What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity

Felipe Fronchetti[1(✉)], Igor Wiese[2], Gustavo Pinto[3], and Igor Steinmacher[4]

[1] University of São Paulo, São Paulo, São Paulo, Brazil
fronchetti@usp.br
[2] Federal University of Technology, Campo Mourão, Paraná, Brazil
igor@utfpr.edu.br
[3] Federal University of Pará, Belém, Pará, Brazil
gpinto@ufpa.br
[4] Northern Arizona University, Flagstaff, Arizona, USA
igor.steinmacher@nau.edu

Abstract. Voluntary contributions play an important role in maintaining Open Source Software (OSS) projects active. New volunteers feel motivated to contribute to OSS projects based on a set of motivations. In this study, we aim to understand which factors OSS projects usually maintain that might influence their new contributors' onboarding. Using a set of 450 repositories, we investigated mixed factors, such as the project age, the number of stars, the programming language used, or the presence of text files that aid contributors (e.g., templates for pull-requests or license files). We used a K-Spectral Centroid (KSC) clustering algorithm to investigated the newcomers' growth rate for the analyzed projects. We could found three common patterns: a logarithmic, an exponential, and a linear growth pattern. Based on these patterns, we used a Random Forest classifier to understand how each factor could explain the growth rates. We found that popularity of the project (in terms of stars), time to review pull requests, project age, and programming languages are the factors that best explain the newcomers' growth patterns.

Keywords: Open Source Software · Newcomers · Attractiveness

1 Introduction

Voluntary contributions play an important role in maintaining Open Source Software (OSS) projects active [29]. This is because OSS projects work in a symbiotic way. While communities need to motivate, engage, and retain new developers to remain sustainable [19], a large, globally distributed community of developers wants to contribute for a variety of reasons, including learning, the necessity to fix a bug, and reputation [13,24,33].

However, as shown before in several studies [28–30], the newcomers face several barriers while joining to OSS projects. This can lead to demotivation and,

Published by Springer Nature Switzerland AG 2019
F. Bordeleau et al. (Eds.): OSS 2019, IFIP AICT 556, pp. 91–103, 2019.
https://doi.org/10.1007/978-3-030-20883-7_9

ultimately, dropouts. Given the importance of the newcomers to the projects and the barriers they face, it is important to study the different aspects of the joining process. As previously stated by Steinmacher et al. [30], joining a project is a complex process composed of different stages and a set of forces that push newcomers towards (motivation and attractiveness) or away (onboarding barriers) from the project. While motivation is something that is usually inherent to the developers, attractiveness is a force that—to some extent—can be managed by the projects.

Some previous studies analyzed project attractiveness by analyzing its relationship with license [22], source code attributes [15,16], and code base [4]. However, the existing literature does not analyze the temporal aspect of the newcomers' joining, nor consider the recent phenomenon of social coding environments and their characteristics, which introduced a more standardized way to contribute [9]. This perspective is important, since, according to Capiluppi and Michlmayr [3] "the success of a project is often related to the number of developers it can attract".

In this paper, we start filling this gap by investigating which projects' characteristics are related to the increase of newcomers growth temporally in OSS projects maintained on GitHub. To achieve that, we selected a set of factors inherently from the OSS projects that might explain the increase of newcomers. We place these factors in context, measuring their effects on 72 weeks of growth of newcomers in 450 OSS projects. Our approach included clustering the OSS projects in terms of newcomers' joining growth, aiming to identify different growth patterns. Based on the patterns identified, we leveraged the Random Forest [2] classifier to measure the effects of the projects' characteristics aiming to explain each pattern.

The contributions of this study include: (i) empirical evidence of different patterns of newcomers growth in OSS projects, adopting a time series analysis; and (ii) identifying the factors that could potentially lead to attraction of newcomers in OSS projects. Ultimately, the results of this work may benefit project maintainers who can get acquainted with ways to make their project's more attractive, creating a more welcoming environment for newcomers.

2 Methodology

To guide our research towards this goal, we designed the following research questions:

- **RQ1. What are the newcomer joining rates in OSS projects?** The answer to this question is relevant to understand if projects receive constant rates of newcomers temporally, or if there are different trends for different projects. In case we find different trends of temporal joining rates, it is worth understanding how these different trends can be classified. By having a comprehensive classification of newcomers joining rate per time, it is possible to go in-depth and explore the reasons for the differences.

– **RQ2. What are the project' factors that may influence the joining rate?** The answer to this question aims to explain what are the factors that may influence different rates of newcomer joining, which may bring to light potential ways to attract more developers to OSS projects.

2.1 Curating the Corpus of OSS Projects

To explore the attractiveness of OSS projects, we retrieved data from 450 OSS projects hosted on the GitHub coding platform. To improve the variety of projects, we first selected the fifteen most popular programming languages used on GitHub [20]. The set of programming languages was composed of C, Clojure, CoffeeScript, Erlang, Go, Haskell, Java, JavaScript, Scala, Objective-C, Perl, PHP, Python, Ruby, and TypeScript. To avoid unmaintained projects, for each selected programming language, we filtered the 30 most starred OSS projects. We followed recent related work that considers stars as a measure of attractiveness in OSS projects [1]. We ended up with 450 OSS projects, written by 15 popular programming languages. The set of sampled projects included:

– SCALA/SCALA: The Scala programming language was released in 2001 and has received more than 31 K commits. Mostly written in Scala.
– DJANGO/DJANGO: High-level web application framework. Release in 2005, it has received more than 26 K commits. Mostly written in Python.
– VIM/VIM: Highly configurable text editor. Released in 1991, it has received more than 9 K commits. Mostly written in C.

As one can see, Fig. 1 suggests that even selecting projects with the highest number of stars, the distribution of stars varies significantly in the dataset, ranging from 6 to 39,990 stars (Q1: 3,648; Median: 10,470; Q3: 18,900). The data was collected using the GitHub API, and it was conducted in October 2018. To access the complete dataset and the source code of the tools used in this research are publicly available in our repository[1].

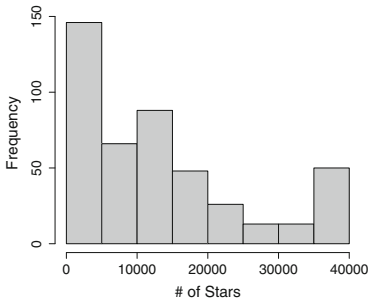


Fig. 1. Stars per project

In our analysis, we collected a set of factors from the selected projects' repositories, including project popularity, maturity, receptivity/welcoming features. We used these factors in our model, and their descriptions are presented in Table 1. We also calculated the Spearman correlation coefficients [12] on the factors in order to remove the strongly correlated ones ($\rho > 0.7$). The number of forks was removed from the list of factors since it has a high correlation with the number of stars.

[1] https://github.com/fronchetti/OSS-2019.

Table 1. Factors extracted from repositories

Factor	Description
Age	Number of years since the repository creation
Main language	Most used programming language
Time to merge	Average of days for pull requests (PR) to be merged
Account type	If the OSS is hosted on an Organization or User account
Domain	The software application domain
# of stars	Number of stars
# of languages	Number of used programming languages
# of integrators	Number of contributors with rights to merge pull requests
Has PR template	Repository has a standard template for new pull requests
Has issue template	Repository has a standard template for new issues
Has license	Repository has the LICENSE file
Has code of conduct	Repository has CODE_OF_CONDUCT file
Has readme	Repository has the README file
Has contributing	Repository has the CONTRIBUTING file
Has wiki	Repository has WIKI+ section

It is important to note that some of these factors are not straightforwardly available in the GitHub repository, which is the case of the domain of the repositories. For this particular factor, we followed the Borges et al. methodology to define projects domain [1], and manually added it by doing a qualitative analysis over the website and documentation of the projects. As an example, we manually evaluate the Linux website[2] to define it as part of the system software domain.

To characterize the project attractiveness temporally, we collected the newcomers' growth rate for each project, considering one week as the observation unit. The growth of newcomers is represented by a time series, which associates the evolution of the number of newcomers with the number of weeks existent in each project (newcomers per week). To this end, we define a newcomer as any contributor that submitted their very first contribution (commit) to the master branch. It is worth mentioning that each contributor was considered a newcomer only once, in the particular week that they submitted their first commit.

2.2 Identifying Growth Patterns

To identify the different newcomers' growth patterns, we clustered the sampled OSS projects growth according to the time series mentioned before. We used the K-Spectral Centroid (KSC) clustering algorithm [32] to create the clusters. The KSC algorithm finds clusters of time series that share distinct temporal

[2] https://www.linux.com/what-is-linux.

patterns, following a similar approach as the used in the classical K-means clustering algorithm [11]. We chose to apply the KSC clustering algorithm because the clustering is performed independently of shifts (i.e., dates) and scale (i.e., volume), focusing rather on the overall shape of the time series [7]. Moreover, the KSC algorithm was applied in well-established papers that follow a similar approach in different contexts [1] and domains [8].

The KSC algorithm requires that all the time series used during the clusterization have the same length. For this reason, we used only the time frame comprising the last 72 weeks of newcomers inflow for each project, considering the date of the dataset creation (October 2018). We use the length of 72 weeks because all projects are at least 72 weeks old. The KSC algorithm also requires the definition of a specific number of k clusters. To decide the best number of k clusters, we used the β_{CV} [17] heuristic. The β_{CV} heuristic is defined as the ratio of two coefficients: variation of the *intracluster* distances and variation of the *intercluster* distances. The smallest value of k after which the β_{CV} ratio remains roughly stable should be selected, as a stable β_{CV} implies that new splits affect only marginally the variations of *intracluster* and *intercluster* distances [8].

Figure 2 presents an association between the β_{CV} ratios and the k clusters for the newcomers time series. When considering the β_{CV} ratios and the number of projects per cluster, we decided to use $k = 3$ clusters in the K-SC algorithm. Note that $k = 4$ clusters could represent a better number of clusters in terms of ratio stability. By testing the clustering algorithm for $k = 3$ and $k = 4$, we found that the results were similar, and two groups could clearly be merged. Moreover, having a small number of projects per cluster would affect the future classification of patterns presented in Sect. 2.3. Thus, we kept with $k = 3$ clusters.

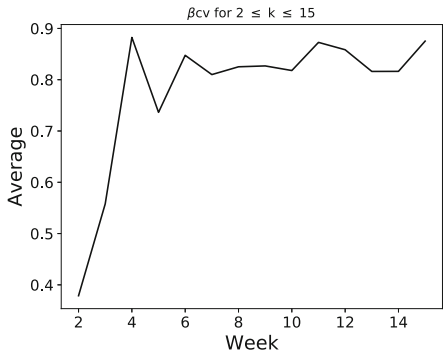


Fig. 2. The β_{CV} ratios

2.3 Identifying Explanations to the Growth Patterns

Our next step was to explore the different newcomers' growth patterns. To achieve this goal, we used a Random Forest [2] classifier to measure the effects of the independent variables (factors) in the explaining of the dependent variable (growth patterns). We selected Random Forest because it is fast [23], robust in the presence of noise and outliers [21], and has a great performance with numerical and categorical data [25].

To develop our model, we used the `RandomForestClassifier` class from the `scikit-learn` framework. The predictors used were the factors defined in Table 1, and the target variables were the growth patterns found. Using a measure called Mean Decrease Impurity (MDI) [14], `RandomForestClassifier` also

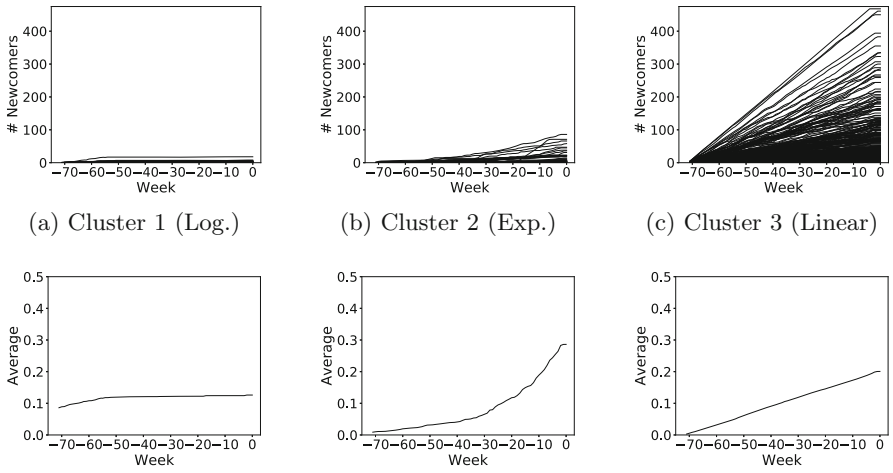
provides a ranking of the most important features in the prediction of the target variables. The higher the score of the feature, the greater is its importance. We used the scores obtained from the classifier to understand the relationship between the factors and the growth patterns. Finally, we measured the effectiveness of the classifier using three commonly used metrics of Machine Learning: Precision, Recall, and F-measure. The code related to the clusterization and classification of the growth patterns are publicly available in our repository (See footnote 1).

Table 2. Description of the clusters

Cluster	Pattern	# repositories	Growth (%)
C1	Logarithmic	71 (15.7%)	12.6%
C2	Exponential	57 (12.6%)	27.6%
C3	Linear	322 (71.5%)	19.9%

3 RQ1. On the Newcomer Joining Rates in OSS Projects

Figure 3 presents the growth patterns found, which we named as *logarithmic*, *exponential* and *linear* growth of newcomers. We chose these names based on the centroids' trends, defined as the average growth of the clusters (also presented in Fig. 3). In Table 2 we present a clusters overview, including the number of repositories per cluster and the percentage increase of newcomers obtained from the



(a) Cluster 1 (Log.) (b) Cluster 2 (Exp.) (c) Cluster 3 (Linear)

(d) Cluster 1 (Centroid) (e) Cluster 2 (Centroid) (f) Cluster 3 (Centroid)

Fig. 3. Growth patterns clusters and centroids.

centroids in 72 weeks. *Linear* growth includes the highest number of repositories (71.5%), with a percentage increase of newcomers of 19.9%. The *Logarithmic* growth is the second one with most repositories (15.7%), but different from the *Linear* growth, it has the lowest increase of newcomers (12.6%). The *Exponential* growth is the less representative cluster, represented by only 12.6% of the repositories. On the other hand, its percentage increase of newcomers is the highest one (27.6%).

To put these results in a better context, we investigated OSS projects for each one of the growth patterns. In the *linear* growth, we perceived that projects DEFINITELYTYPED/DEFINITELYTYPED, RAILS/RAILS, and SYMFONY/SYMFONY, follow this tendency from the very beginning. We perceived a similar trend when analyzing the projects that fit on the *logarithm* growth. The projects ALPACALANG/ALPACA and CHAPLINJS/CHAPLIN are interesting samples because both are active projects that received their first newcomer only after the first two months of analysis. On the other hand, the projects IONIC-TEAM/IONIC, SYNRC/N2O, and DRKLO/TELEGRAM are the ones who fit in the *exponential* growth. In particular, project Ionic grew from 25 to 45 newcomers in only ten weeks of activity.

RQ1 Summary. Three different growth patterns represent the entry of newcomers into OSS projects: Linear, exponential and logarithmic. Linear growth represents the majority of repositories with an intermediate growth; exponential growth represents the smallest number of repositories, but holds the highest growth of newcomers; and logarithmic growth represents an intermediate number of repositories, but has the lowest growth among the three patterns.

4 RQ2: On the Factors that May Influence the Joining Rate

After running the Random Forest classifier and calculating the MDI for the factors in our prediction model, we rank the projects' factors. Table 3 presents factors ordered by importance in predicting of newcomers growth patterns. As one could see, the highest score is the number of stars. This finding is particularly interesting because, contrary to well-known beliefs that suggest that newcomers may be more tempted to contribute to OSS projects that are written in a programming language that they are more familiar with, than a popular one. Time to merge appears next in the top factors explaining the newcomers' growth rate. This is interesting, because it shows a relationship between a newcomers onboarding and the good practice of giving timely review, feedback, and closing pull requests. Completing the list of the top factors (with scores higher than 0.10), we have factors that are intrinsically related to the project, such as age and the number of programming languages used.

Moreover, the presence of text files such as the CONTRIBUTING file, the LICENSE file, and the CODE OF CONDUCT file, which are even recommended

Table 3. Ranking of the most important factors

Ranking	Factor	Score
1	# of stars	0.1753
2	Time to merge	0.1535
3	# of languages	0.1278
4	Age	0.1027
5	# of integrators	0.0995
6	Main language	0.0946
7	Domain	0.0708
8	Has contributing	0.0396
9	Has wiki	0.0308
10	Has issues template	0.0260
11	Owner type	0.0252
12	Has license	0.0236
13	Has PR template	0.0183
14	Has code of conduct	0.0118

as community best practices[3], are among the worst ranked factors in our model (they scored 0.0396, 0.0236, and 0.0118, respectively). Still, having issues and pull requests templates, which are also recommended to welcome newcomers, presented very low scores (0.0260 and 0.0183, respectively).

We also investigated the effectiveness of the classifier in predicting the growth patterns. We used three metrics from `scikit-learn`[4] to investigate the effectiveness: Precision, Recall, and F-measure [26]. Precision measures the correctness of the classifier in predicting growth patterns. Recall measures the effectiveness of the classifier in identifying the growth patterns. F-measure is the harmonic mean of precision and recall. Table 4 shows the metrics' results divided by cluster, along with an overall result based on the micro-average of each metric [27].

Table 4. Precision, recall, and f-measure of the classification model. Divided by clusters, and an overall.

Growth Pattern	Precision	Recall	F-measure
Logarithmic	0.44	0.19	0.27
Exponential	0.33	0.06	0.10
Linear	0.75	0.95	0.84
Overall	0.72	0.72	0.72

[3] https://opensource.guide.
[4] https://scikit-learn.org.

In general, the Random Forest classifier obtained a significant performance, with a micro-average of 72% for precision, recall, and F-measure. The *Linear* growth group presented the highest results, and almost all its instances were correctly classified (Precision: 75%, Recall: 95%, F-measure: 84%). On the other hand, we could not observe good results for the *Logarithmic* and *Exponential* groups. Only 6% of the *Exponential* instances were identified correctly (Precision: 33%, Recall: 6%, F-measure: 10%). The bad results may be justified by the number of instances analyzed (We only used 450 projects), the number of instances per growth group (since 71.5% of the instances belong to the *Linear* pattern), and the characteristics used during the prediction (Other characteristics may also affect the distinction of patterns). However, to understand these differences accurately, a broader study is needed.

RQ2 Summary. Popularity of the project (in terms of stars), time to review pull requests, and project characteristics like age and programming languages are the factors that best explain the newcomers' growth patterns. In addition, GitHub recommended community standards (https://github.com/github/opensource.guide/community) have a lower influence on the observed growth patterns.

5 Related Work

Several studies focus on how newcomers join OSS projects [18,30,31]. Von Krogh et al. [31] propose a joining script for developers who want to take part in a project. Similarly, Nakakoji et al. [18] proposed an onion based structure (the onion patch) to explain the OSS joining process. Steinmacher et al. [30] proposed a joining model, in which they represent motivation and attractiveness as forces that influence outsiders to become newcomers to OSS project. In this study, we focus on the characteristics of the project that may explain the attraction of newcomers in terms of temporal onboarding growth.

The attractiveness topic had also been previously studied. For example, Santos et al. [22] defined a theoretical cause-effect model for attractiveness to OSS projects, proposing its typical causes (license type, intended audience, type of project, development status), indicators (hits, downloads, members), and consequences (number of open tasks, time for task completion). They found that projects for end-users and developers have higher attractiveness, that application domain impacts attractiveness, and that projects licensed under most restrictive licenses tend to be less attractive. These results contradict Colazo and Fang's [5] results, which analyzed 62 projects from SourceForge and found that restrictively licensed projects are more attractive to volunteer OSS developers.

From a different perspective, Meirelles et al. [16] applied the same model as Santos [22], inserting source code metrics as a typical attractiveness causes. They observed that structural complexity and software size (lines of code and number of modules), indicating that structural complexity negatively influences attractiveness, whereas software size positively influences it. Chengalur-Smith et al. [4]

analyzed whether codebase size, project age, and niche size (a measure borrowed from ecology) influenced project attractiveness, finding that these three characteristics indeed influence the project's ability to attract and retain developers.

Although the attractiveness topic has been explored from different perspectives, the studies mentioned to not consider temporal growth trends in newcomer's onboarding. Moreover, the aforementioned studies do not analyze the attractiveness after the social coding environments become commonplace in OSS development. One exception is the paper by Gupta et al. [10], who analyzed how the adoption of continuous integration impacts developer attraction. However, they analyzed this single intervention, without considering any other project characteristic.

6 Limitations

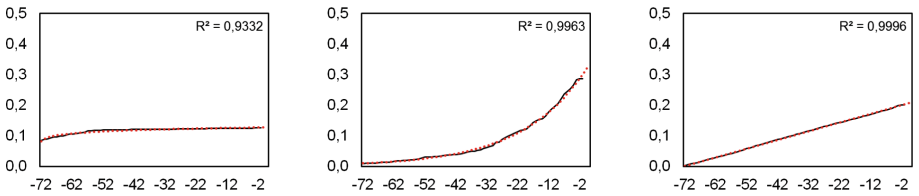
In a study such as this, there are always many limitations and threats to validity. First, we considered only a limited number of attributes to explain the phenomenon of newcomers onboarding growth. Although we understand that different attributes and different ways to compute the factors should be used, we focused on factors that cover project popularity, maturity, skills required, receptivity/welcoming features.

Second, we focused only on GitHub OSS projects—the largest OSS hosting environment to date —, which means that our findings may not generalize to other platforms with different contributing characteristics. Moreover, we diversified our sample including 30 projects for each of the 15 most popular programming languages in GitHub, which naturally increases the diversity of our sample. However, our focus on the most popular OSS projects may have influenced factors such as the "# of Stars". Nevertheless, as depicted at Fig. 1, the selected OSS are greatly diverse, when it comes to the number of stars. Still, although large, our dataset clearly does not comprehend the whole universe of OSS projects available. Also, we did not distinguish spare time contributors from employees hired by a software company to contribute to OSS. We are aware that they may have different contributing behaviors [6], but a comprehensive analysis of their joining rate is left this for future work.

Third, the decisions regarding the observation unit (in our case, one week) can also be seen as a limiting factor. However, it is important to note that, when exploring our data, and we found similar behavior for higher time windows. Similarly, we also considered that the use of 72 weeks to represent the joining rate of newcomers may not be sufficient. However, this is a limitation of the KSC algorithm, which requires an equivalent time window for all analyzed series.

Finally, we understand that only three clusters may not represent the diversity of joining rates found in OSS projects. However, we tried to overcome this limitation by using the β_{CV} heuristic. Yet, one might argue that our cluster may not strictly follow the function curves we indicated (exponential, linear, and logarithm). However, to make sense of this, we investigated the coefficients that indicate a strong correlation with the centroids and the growth patterns.

Figure 4 shows this relationship. In each figure, there is a black line (the centroid line) and a red dotted line (the trend line that represents the function curve). As indicated, both lines follow roughly the same curve (the closer R2 is to 1 indicates the stronger is the correlation).



(a) Cluster 1 (Log.) (b) Cluster 2 (Exp.) (c) Cluster 3 (Linear)

Fig. 4. Comparing the growth patterns with the default curves. (Color figure online)

7 Conclusion

In this paper, we investigated the projects' characteristics that may explain the different patterns of newcomers growth in OSS projects. Through a sequence of quantitative and statistical analyses based on data mined from 450 OSS projects, we were able to uncover several so far unknown behaviors of OSS projects. For instance, we perceived that there are three main onboard growth patterns (namely a logarithm growth pattern, a linear growth pattern, and an exponential growth pattern). Moreover, we also shed some light on the factors that might encourage newcomers to onboard on the OSS projects. The Top-3 ranked factors were: the number of stars, the time to merge a pull-request, and the number of programming languages used. For future work, we plan to leverage qualitative analysis to better understand what external reasons (e.g., new release, recently open sourced, the first page on HackerNews, etc.) might lead some projects to an exponential growth of newcomers.

Acknowledgment. This work is partially supported by CNPq (#430642/2016-4 and #406308/2016-0), Fundação Araucária and FAPESP (#2015/24527-3).

References

1. Borges, H., Valente, M.T.: What's in a GitHub star? Understanding repository starring practices in a social coding platform. J. Syst. Softw. **146**, 112–129 (2018)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Capiluppi, A., Michlmayr, M.: From the cathedral to the bazaar: an empirical study of the lifecycle of volunteer community projects. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds.) OSS 2007. ITIFIP, vol. 234, pp. 31–44. Springer, Boston, MA (2007). https://doi.org/10.1007/978-0-387-72486-7_3

4. Chengalur-Smith, I.N., Sidorova, A., Daniel, S.L.: Sustainability of free/libre open source projects: a longitudinal study. J. Assoc. Inf. Syst. **11**(11), 657–683 (2010)
5. Colazo, J., Fang, Y.: Impact of license choice on open source software development activity. J. Am. Soc. Inf. Sci. Technol. **60**(5), 997–1011 (2009). https://doi.org/10.1002/asi.v60:5
6. Dias, L.F., Steinmacher, I., Pinto, G.: Who drives company-owned OSS projects: internal or external members? J. Braz. Comp. Soc. **24**(1), 16:1–16:17 (2018)
7. Figueiredo, F.: On the prediction of popularity of trends and hits for user generated videos. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 741–746. ACM (2013)
8. Figueiredo, F., Almeida, J.M., Gonçalves, M.A., Benevenuto, F.: On the dynamics of social media popularity: a YouTube case study. ACM Trans. Internet Technol. (TOIT) **14**(4), 24 (2014)
9. Gousios, G., Pinzger, M., Deursen, A.: An exploratory study of the pull-based software development model. In: 36th International Conference on Software Engineering, ICSE 2014, pp. 345–355 (2014)
10. Gupta, Y., Khan, Y., Gallaba, K., McIntosh, S.: The impact of the adoption of continuous integration on developer attraction and retention. In: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pp. 491–494, May 2017
11. Hartigan, J.A.: Clustering algorithms (1975)
12. Hauke, J., Kossowski, T.: Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. Quaestiones geograph. **30**(2), 87–93 (2011)
13. Ke, W., Zhang, P.: The effects of extrinsic motivations and satisfaction in open source software development. J. Assoc. Inf. Syst. **11**(12), 784–808 (2010)
14. Louppe, G., Wehenkel, L., Sutera, A., Geurts, P.: Understanding variable importances in forests of randomized trees. In: Advances in Neural Information Processing Systems, pp. 431–439 (2013)
15. Maalej, W., Happel, H.J., Rashid, A.: When users become collaborators: towards continuous and context-aware user input. In: Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA 2009, pp. 981–990. ACM (2009)
16. Meirelles, P., Santos, C., Miranda, J., Kon, F., Terceiro, A., Chavez, C.: A study of the relationships between source code metrics and attractiveness in free software projects. In: 2010 Brazilian Symposium on Software Engineering, SBES 2010, pp. 11–20. IEEE (2010)
17. Menasce, D.A., Almeida, V.A.: Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice Hall PTR, Upper Saddle River (2002)
18. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., Ye, Y.: Evolution patterns of open-source software systems and communities. In: Proceedings of the International Workshop on Principles of Software Evolution, IWPSE 2002, pp. 76–85. ACM, New York (2002)
19. Qureshi, I., Fang, Y.: Socialization in open source software projects: a growth mixture modeling approach. Organ. Res. Methods **14**(1), 208–238 (2011)
20. Ray, B., Posnett, D., Filkov, V., Devanbu, P.: A large scale study of programming languages and code quality in GitHub. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 155–165. ACM (2014)

21. Robnik-Šikonja, M.: Improving random forests. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 359–370. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30115-8_34
22. Santos, C., Kuk, G., Kon, F., Pearson, J.: The attraction of contributors in free and open source software projects. J. Strategic Inf. Syst. **22**(1), 26–45 (2013)
23. Segal, M.R.: Machine learning benchmarks and random forest regression (2004)
24. Shah, S.K.: Motivation, governance, and the viability of hybrid forms in open source software development. Manag. Sci. **52**(7), 1000–1014 (2006)
25. Shi, T., Horvath, S.: Unsupervised learning with random forest predictors. J. Comput. Graph. Stat. **15**(1), 118–138 (2006)
26. Sokolova, M., Japkowicz, N., Szpakowicz, S.: Beyond accuracy, f-score and ROC: a family of discriminant measures for performance evaluation. In: Sattar, A., Kang, B. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1015–1021. Springer, Heidelberg (2006). https://doi.org/10.1007/11941439_114
27. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. **45**(4), 427–437 (2009)
28. Steinmacher, I., Conte, T., Gerosa, M.A., Redmiles, D.: Social barriers faced by newcomers placing their first contribution in open source software projects. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2015, pp. 1379–1392. ACM (2015)
29. Steinmacher, I., Conte, T.U., Treude, C., Gerosa, M.A.: Overcoming open source project entry barriers with a portal for newcomers. In: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, pp. 273–284. ACM, New York (2016)
30. Steinmacher, I., Gerosa, M.A., Redmiles, D.: Attracting, onboarding, and retaining newcomer developers in open source software projects. In: Proceedings of the Workshop on Global Software Development in a CSCW Perspective, CSCW 2014 Workshops (2014)
31. von Krogh, G., Spaeth, S., Lakhani, K.R.: Community, joining, and specialization in open source software innovation: a case study. Res. Policy **32**(7), 1217–1241 (2003)
32. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 177–186. ACM (2011)
33. Ye, Y., Kishida, K.: Toward an understanding of the motivation open source software developers. In: 25th International Conference on Software Engineering, ICSE 2003, pp. 419–429. IEEE Computer Society, Washington (2003)