Long-Term Active Integrator Prediction in the Evaluation of Code Contributions

Jing Jiang, Fuli Feng, Xiaoli Lian, Li Zhang
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
{jiangjing, lily}@buaa.pku.edu.cn, fulifeng93@gmail.com, lxl_tuizi@hotmail.com

***Abstract*—In open source software (OSS) projects, integrators are given high-level access to repositories so that they could maintain and manage projects. Although integrators play a critical role in evaluating code changes for OSS projects, they may be short-term active. Long-term active integrators keep in evaluating code update submission and managing responses from contributors. In order to survive and succeed, OSS projects need to attract and retain long-term active integrators. To assist OSS projects to retain active integrators, we propose a method called LTAPredict to predict whether integrators will be long-term active in the evaluation of code contributions. LTAPredict collects activity data of integrators, extracts a rich set of features, and makes prediction via machine learning techniques. We perform experiments on 37 popular projects, containing a total of 1,073 integrators. Results show that based on the Decision Tree, LTAPredict achieves the accuracy as 0.829, the precision as 0.81, the recall as 0.827 and the F1 as 0.818. Meanwhile, we evaluate the feature importance to identify the most significant indicators of long-term active integrators. We observe that whether integrators becoming long-term active is associated with the number of active months and social distance with contributors in their first year as integrators. These findings assist OSS projects to identify potential long-term active integrators and adopt better strategies to retain them in the evaluation of code contributions.**

***Keywords*-Long-term active integrator; Code contributions; Open source software**

I. Introduction

In open source software (OSS) projects, contributors fix bugs or improve functions, and then submit these code changes to original projects [1]. Integrators are granted the privilege of directly committing codes to source code repositories [2], [3], [4]. Integrators are responsible for reviewing code updates submitted by contributors, and deciding whether or not merge these code changes into repositories. Integrators not only merge satisfactory codes into repositories, but also discuss with contributors and encourage them to keep in submitting codes.

Although integrators play a critical role in evaluating code changes for OSS projects, they may be short-term active in the evaluation of code contributions. Due to the principle of voluntary participation, integrators always have the freedom to decide their activities and even leave the community [5]. Furthermore, some integrators may focus on writing codes by themselves, and seldom spend time in evaluating code contributions from others. Long-term active integrators keep in evaluating code modifications and merging satisfactory codes into repositories. Long-term active integrators in the evaluation are crucial for projects success, because their participation in a long period keeps the sustainable development for OSS projects. In addition, if integrators always change, it is hard for contributors to know evaluative criteria, which may further discourage volunteers to make contributions [6].

Previous works have explored how new participants join projects [1], make contributions [7], [8], enter the circle of trust [9] and finally become integrators [10]. Some other works have studied factors which impact decisions in the evaluation of code contributions [2], [3]. However, it remains unknown whether integrators are active in a long time or not, and what affect their decisions. Thus there is a need for a comprehensive analysis of long-term active integrators in the management of code contributions.

In this work, we explore what make long-term active integrators in open source project-hosting site GitHub [11]. GitHub is one of the world's largest open source communities, and it provides the pull request model for contributors to submit code updates [2] We mainly study pull requests to understand integrators' contributions in code evaluation. We use GitHub API to collect data, and obtain information of 37 popular projects and their 1,073 integrators. We study active periods of integrators in the management of code contributions. Based on analysis results, we define integrators as long-term active if they have joined projects for more than 1 year and keep active in evaluating code contributions in more than 30% of months.

We propose a method called LTAPredict to predict whether integrators will be long-term active. LTAPredict extracts various kinds of features to comprehensively describe integrators, including willingness and capacity, environment, experience and social status. LTAPredict uses these features and makes prediction via machine learning techniques. Experimental results show that based on the Decision Tree, LTAPredict achieves the accuracy as 0.829, the precision as 0.81, the recall as 0.827 and the F1 as 0.818. Meanwhile, we evaluate the feature importance to identify the most significant indicators of long-term active integrators. We observe that whether integrators becoming long-term active is associated with the number of active months and social distance with contributors in their first year as integrators. These findings assist OSS projects to identify potential long-term active integrators and adopt better strategies to retain them in the evaluation of code

DOI reference number: 10.18293/SEKE2016-030

contributions.

II. Background and Datasets

In this section, we begin by providing background information about contribution evaluation process in GitHub. Then, we introduce how our datasets are collected. Finally, we report statistics of our datasets.

A. Contribution Evaluation Process

In GitHub, contributors fork repositories, use codes as their own and make changes [2]. Contributors submit pull requests when they want to merge their changes into main repositories. Integrators inspect code changes and evaluate potential contributions. Integrators have several options towards pull requests: If pull requests are deemed satisfactory, they merge code changes into main repositories; otherwise, they may directly reject and close them, or ask contributors to make updates. Integrators may also ignore code contributions, and leave pull requests open. For each pull request, an issue is opened automatically, where integrators and other interested users exchange comments.

Integrators, who evaluate the quality of code change submission, decide project evolution and play a critical role for OSS projects. In order to becoming integrators, developers need to successfully pass the evaluation of technical contributions and social interactions [7], [9], [10]. After becoming integrators, they are given high-level access to repositories, and have the responsibility of evaluating code update submission and managing responses from contributors. Although integrators play a critical role in evaluating code changes for OSS projects, they may be short-term active in the evaluation of code contributions. The principle of voluntary participation attracts a lot of people to join the OSS community, while it also means that people always have the freedom to decide their activities or even leave the community [5]. Furthermore, some integrators may focus on writing codes by themselves, and seldom spend time in evaluating code contributions from others.

B. Data Collection

GitHub provides access to its internal data stores through an API [1]. It allows us to access a rich collection of OSS projects, and provides valuable opportunities for research. We gather information from GitHub API and create a dataset of projects and integrators. The process of data collection is as follow:

We obtain project lists from MSR 2014 Mining Challenge MySQL Dataset [12]. This dataset includes 90 popular software projects for top programming languages in GitHub. We focus on popular and active projects, because they may need integrators. Small projects often have few integrators, and their project owners can manage projects by themselves.

Next, we downloaded historical information of 90 projects in July 2014, including their pull requests. Then the following criteria is applied to exclude projects from the initial selection: Firstly, projects should have at least one event of activities within 1 month prior to data collection (July 2014), so as to avoid inactive projects. Secondly, projects should be created at least two years prior to data collection. It ensures that projects have more than two years of historical information. We are interested to explore how their integrators behave as time goes on. Finally, projects should have at least 300 pull requests. For our analysis, we use the evaluation process of pull requests to identify active integrators. We choose projects which use pull requests as the important method for code contributions in GitHub.

After selection, our sample includes 37 projects. We make basic analysis of projects in our dataset. 8 projects were created earlier than July 2009, and have histories longer than 5 years; 20 projects have histories longer than 4 years, and 32 projects have histories longer than 3 years. Since GitHub was founded in April 2008 [2], these projects have long histories in GitHub and provide great opportunities to explore evolution of integrators. Our dataset includes projects written in representative languages, such as PHP, Ruby, Python, C, C++, JavaScript and Scala [13].

We collect detailed information of pull requests from all developers in 37 projects through GitHub API. Then we collect project integrators from pull requests. GitHub does not provide the exact promotion time of integrator, and we use the first activity time to estimate the promotion time and determine the integrator. If a developer firstly merges the pull request, or closes a pull request from another developer, then this developer is considered as an integrator, and the promotion time is estimated by this first activity time. We exclude activity records before the first activity time, so as to make sure that the user has already been promoted as the integrator. Next, we extract pull request records for integrators. Integrators take different kinds of actions to handle pull requests, such as accepting, rejecting, closing or discussing pull requests. We extract these activities from our datasets, and build records about *when* pull requests are handled by *which* integrators with *which* actions. GitHub is a social coding site [11], and allows users to attract followers. We also collect follower and following relationships of integrators. Finally, our dataset includes 1,073 integrators and their 104,910 records of pull request evalution.

C. Basic Statistics

In this subsection, we report basic statistics of our datasets. The integrator's age is defined as the number of months between the integrator's promotion time and data collection. For every active integrator, we compute the number of active months after the promotion, divided by the integrator's age. In our computation, we exclude integrators who join projects less than 1 year before our data collection. This is because that the time is too short to study their characteristics. Figure 1 shows complementary cumulative distribution function (CCDF) of the percentage of active months for active integrators. 64.1% of integrators have the percentage of active months larger

[1] http://developer.github.com/v3/

[2] https://github.com/blog/40-we-launched

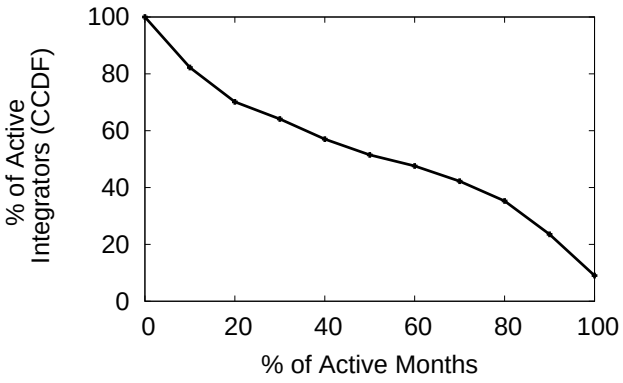

Fig. 1. Percentage of active months

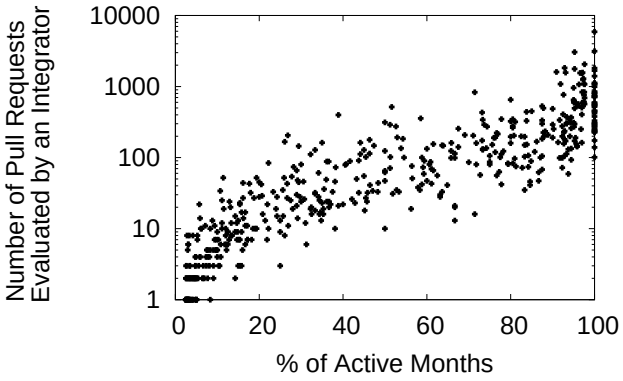

Fig. 2. The relationship between active periods and productivity

than 30%, while other 35.9% of active integrators have the percentage of active months less than 30%. The majority of active integrators keep in evaluating code contributions in a long time, while a minority of integrators occasionally handle pull requests.

We study the relationship between the number of active months and the amount of pull requests evaluated by the integrator. Figure 2 shows the percentage of active months versus the number of pull requests evaluated by the integrator. When an integrator has the larger percentage of active months, this integrator is likely to evaluate more pull requests. The Pearson correlation between the percentage of active months and the number of pull requests is as high as 0.51. The productivity in contribution evaluation is strongly correlated with the active time.

We consider the definition of long-term active and short-term active integrators in the evaluation of pull requests. In fact, there is no standard for the definition, and different people may have various attitude. The definition of long-term active integrators is mainly used to study impact factors of their activities. We also try other thresholds of the percentage of active months, and observe similar results of feature importance and prediction results. Figure 1 shows that 64.1% of active integrators have the percentage of active months larger than 30%. We choose 30% as the threshold to classify active integrators. According to active periods and productivity, we classify active integrators into three categories: *short-term active*, *long-term active* and *unclassified*. Firstly, integrators are *long-term active* if they have joined projects for more than 1 year, and keep active in evaluating code contributions in more than 30% of months. Joining projects at least 1 year before data collection ensures that our data is enough for statistical analysis. Secondly, integrators are *short-term active* if they have joined projects for more than 1 year, but keep active in evaluating code contributions in less than 30% of months. Finally, if integrators join projects less than 1 year before our data collection, there is not enough data to make analysis. These integrators are *unclassified*, and they are excluded in our research and left for future research. Figure 2 shows that the productivity has strong correlation with active periods. Therefore, the percentage of active months is enough to measure the productivity of integrators.

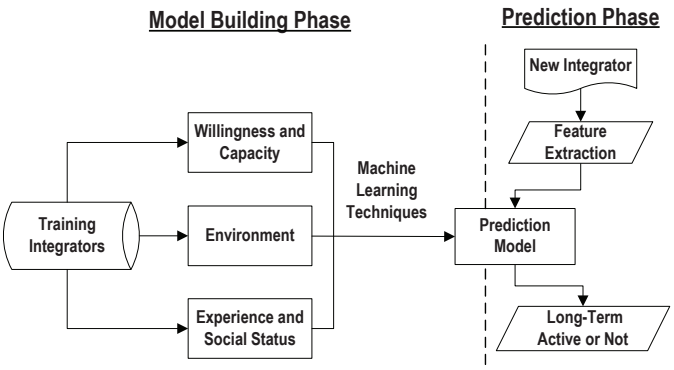

Fig. 3. Overall framework of our method LTAPredict

III. LTAPredict: A Method to Predict Long-Term Active Integrators

In this section, we describe our method LTAPredict to predict long-term active integrators. Figure 3 presents the overall framework of LTAPredict. The entire framework contains two phases: a model building phase and a prediction phase. In the model building phase, our goal is to build a model from training datasets. In the prediction phase, the model is used to predict whether an integrator will be long-term or short-term active.

As described in the subsection II-B, activity data of integrators is firstly collected. Next, we generate features for our analysis based on prior literature about pull request acceptance [2], [3], long-term contributor [1], social coding [11], [14] and committer promotion [9], [10]. We split selected features into three categories, including willingness and capacity, environment, experience and social status. We make basic statistics of these features in Table I. We describe detailed definitions and why we choose these features in subsections III-A, III-B and III-C. Then we use different machine learning techniques to build the prediction model, as described in the subsection III-D. In the prediction phase, we use the LTAPredict to predict whether an integrator will be long-term or short-term active.

TABLE I
SELECTED FEATURES AND BASIC STATISTICS OF ACTIVE INTEGRATORS

Feature	5%	Mean	95%
Willingness and Capacity			
initial_active_month	12	7.19	1
handle_time (day)	49.83	13.8	0.28
num_comments	15	6.35	1
Environment			
total_pulls	2,872	707.95	17
Experience and Social Status			
age (day)	1448.83	728.71	91.23
social_distance	0.42	0.09	0
num_follower	1680	326.77	1
num_following	76	17.61	1

A. Willingness and Capacity

The probability for a new developer to become a long-term contributor may be influenced by the willingness and capacity [15]. Willing and capacity are important factors to decide contributors' activeness in future, and they may also influence integrators' activities. We use three features to measure integrators' willingness and capacity.

Initial Active Month: Gharehyazie et al. found that the number of patches a contributor submitted in early months was the important factor to predict the promotion of integrators [10]. We wonder whether activity frequency is also important after developers become integrators. We consider several features to describe activity frequency of integrator in early months: the number of pull requests the integrator evaluates in the first year as an integrator, the number of pull requests the integrator evaluates divided by the total number of pull requests in the first year, and the number of active months in the first year. To check whether these features are sufficiently independent, we leverage Spearman's rank correlation coefficient [16] (ρ). We observe that ρ between any two features is more than 0.8, and these features are highly correlated. Therefore, our analysis uses the number of active months in the first year as an integrator.

Handle Time: After a contributor submits the pull request, it takes some time for an integrator to evaluate the contribution and give feedback. We compute the average interval time between the pull request's submission and the integrator's first responding. It describes whether the integrator handles pull requests quickly or slowly. Note that the integrator may take several actions towards a pull request, such as leaving comments and then closing the pull request. We use the first action in this feature. The interval time between the pull request's submission and the integrator's last responding is not used, because it is highly correlated with that of first responding.

Num Comments: Marlow et al. observed that uncertain contributions required explanation and discussion [17]. Pull requests with many of comments may be more complicated to evaluate [3], and cost integrators additional time to negotiate with submitters. Due to the high degree of uncertainty, integrators may be unwilling to evaluate these contributions. To measure the level of discussion, we compute the average number of comments in pull requests handled by the integrator.

B. Environment

Previous work showed that the project environment at the joining time had obvious impact on the contributors' behavior [15]. We explore whether project environment influences integrators' activities.

Total Pulls: To study environment impact on the integrator, we count the total number of pull requests accumulated in the project, when the developer becomes an integrator. This feature has strong correlation with other measures, including the project's age [10], current size of core team (number of active integrators) [2], [3] and number of contributors who have submitted pull requests before. Therefore, the total number of pull requests is a representative feature to describe the project environment.

C. Experience and Social Status

Sinha et al. observed that the prior experience influenced the probability of the promotion from contributors to integrators [9]. Tsay et al. found that the contributors's social status affected the probability of pull request acceptance [3]. We wonder whether experience and social status are also associated with the likelihood of integrators becoming long-term active. We use four features to measure the experience and social status.

Age: We measure the age of the integrator when he or she firstly handles the pull request in the project. It is calculated as the interval time between the integrator's registration in GitHub and the first pull request evaluation in this project. It measures whether the integrator is new or old in GitHub.

Social Distance: GitHub is a social coding site, and integrates social media functionality with code management tools [11]. Users follow interesting developers and build social connections. Tsay et al. found that pull request acceptance was influenced by social connections between integrators and submitters [3]. Social connections may also influence activeness of integrators. For pull requests handled by the integrator, we compute the percentage of submitters who directly follow this integrator.

Num Follower: This feature is the number of followers the integrator has at the data collection time. The number of followers shows the status of the integrator in the OSS community [3], [14]. A large number of followers means that the integrator is more attractive and influential. We observe the number of followers is highly correlated with the number of repositories owned by the integrator. The number of follower also indicates how actively the integrator creates projects.

Num following: This feature is the number of developers followed by the integrator. It describes whether the integrator actively builds social connections with others.

D. Prediction

We use machine learning techniques to make prediction. In training datasets, we know whether the integrator becomes long-term active or not. Each integrator belongs to a category, namely long-term active or short-term active. Several machine learning algorithms are known to perform well and have been

TABLE II
MEAN PERFORMANCE ON 10-FOLD CROSS VALIDATION

Model	AUC	ACC	PRE	REC	F1
Decision Trees	0.876	0.829	0.81	0.827	0.818
Logistic Regression	0.899	0.808	0.846	0.717	0.776
Neural Network	0.882	0.792	0.796	0.743	0.769
Random Forests	0.848	0.82	0.789	0.835	0.811
Support Vector Machines	0.862	0.817	0.817	0.772	0.794
k-Nearest Neighbor	0.752	0.728	0.795	0.557	0.655

used in previous works [2], [18]. We run training datasets and build the prediction model of LTAPredict through machine learning classifiers, including Decision Trees (DT), Logistic Regression (LR), Neural Network (NN), Random Forests (RF), Support Vector Machines (SVM) and k-Nearest Neighbor (kNN). We implement LTAPredict on top of the software Rapidminer Toolkit [3]. According to project requirements and experimental results, a suitable machine learning classifier can be chosen for LTAPredict.

For a new integrator, we use LTAPredict to predict the category which the integrator will belong to. If the category is long-term active, this integrator is predicted to be long-term active in the evaluation of code contributions; otherwise, the integrator is predicted to be short-term active.

IV. EXPERIMENTS AND RESULTS

In this section, we evaluate our method LTAPredict. The experimental environment is a windows server 2012, 64-bit, Intel(R) Xeon(R) 1.90 GHz server with 24GB RAM. We consider 5 metrics to evaluate experimental results, including Precision (PRE), Recall (REC), F-measure (F1), Accuracy (ACC) and Area under the receiver operating characteristic curve (AUC). We use these metrics, because they have been used in the acceptance prediction of code contributions [2]. In addition, 10-fold cross validation is applied on our experiments and every value of measurement on performance is average of that of each iteration.

Table II illustrates the mean performance of LTAPredict based on different classifiers. *Based on the Decision Tree, LTAPredict achieves the accuracy as 0.829, the precision as 0.81, the recall as 0.827 and the F1 as 0.818.* Except the k-Nearest Neighbor, LTAPredict based on other classifiers gets the AUC over 0.8, and LTAPredict based on the Logistic Regression achieves the AUC as 0.899. LTAPredict based on k-Nearest Neighbor also achieves the lowest value of ACC. Based on different classifiers, LTAPredict maintains PRE at about 0.8 with a very small fluctuation. When REC is considered, LTAPredict based on the Random Forests performs the best. LTAPredict based on the Decision Trees has the F1 as 0.818, and LTAPredict based on the Random Forests has the F1 as 0.811, which are better than other classifiers. Different machine learning classifiers bring various results for LTAPredict. In practice, a specific machine learning classifier can be chosen, according to project requirements and experimental results.

Previous work [2] uses performance decrease to identify important indicators of pull request acceptance. We also use performance decrease to estimate the importance of a feature, when it is filtered out by the select attribute operator. Since we care most on the percentage of real long-term active integrators identified by the classifier, we choose LTAPredict based on the Random Forests, which achieves the highest REC. Then we use the REC decrease to rank feature importance.

[3]http://rapidminer.com/

TABLE III
PERFORMANCE DECREASE ON 10-FOLD CROSS VALIDATION, WHEN EACH ATTRIBUTE IS FILTERED OUT

Attribute	AUC	ACC	PRE	REC	F1
social_distance	-0.012	-0.165	-0.049	-0.439	-0.295
initial_active_month	-0.105	-0.2	-0.142	-0.435	-0.317
num_follower	-0.023	-0.127	-0.032	-0.338	-0.211
age	-0.046	-0.106	-0.026	-0.278	-0.167
num_comments	-0.027	-0.118	-0.057	-0.27	-0.174
num_following	-0.028	-0.09	-0.024	-0.232	-0.137
handle_time	-0.017	-0.051	-0.14	-0.148	-0.144
total_pulls	-0.02	-0.047	0.006	-0.148	-0.074

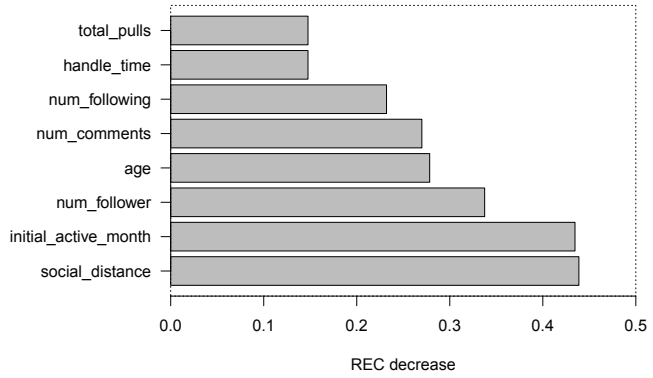

Fig. 4. Feature importance

Based on the Random Forests, the LTAPredict's performance decrease caused by absence of each attribute is illustrated in Table III. It shows that REC will drop significantly if any feature is absent and some of them would bring disastrous influence on performance of LTAPredict. Rank of feature importance can be seen in Figure 4. It is obvious that an almost half drop of REC could happen without either social_distance or initial_active_month. *Therefore, whether an integrator will keep handling pull-requests is mostly decided by the social distance with the contributors and the number of active months in the first year.* In Figure 4, social_distance, number_follower and age cause great REC decrease, which shows that the experience and social distance are strongly associated with the likelihood of integrators becoming long-term active.

V. THREATS TO VALIDITY

Threats to internal validity relate to experimenter bias and errors. Firstly, there is not any standard definition of long-term active integrators. As a result, we define integrators as long-term active, if they have joined projects for more than 1 year and keep active in evaluating code contributions in more than 30% of months. Then we use this definition to identify long-term active integrators and find dominant features. We also try other thresholds of the percentage of active months and obtain similar results of feature importance and prediction

results. The choice of the percentage of active months has few impacts on our results. Secondly, we use integrators' activities in the first year to make prediction. In future work, we will explore whether integrators' activities in a shorter period can be used to make prediction. Thirdly, we use the first activity time of contribution evaluation to estimate the promotion time. If the integrator keeps inactive for a long time after the promotion, the actual promotion time is much earlier than the estimation time. In this case, the estimation time is not correct, but it has few impacts on results. This is because we mainly study activities in code evaluation, and this integrator has no activities of code evaluation between the actual promotion time and estimation time.

Threats to external validity relates to the generalizability of our study. Our empirical findings are based on open source projects in GitHub, and it is unknown whether our results can be generalized to other OSS platforms. In the future, we plan to study a similar set of research questions from other platforms such as Bitbucket, and compare their results with our findings in GitHub.

VI. Related Works

Some previous works explored how new participants joined projects, made contributions and became core members finally [1], [7], [8], [9], [10], [19], [20]. Herraiz et al. found that volunteers tended to follow a step-by-step joining process, while hired developers were usually promoted as integrators suddenly [7]. Gharehyazie et al. observed that the amount of two-way communications a person participated in, was a significant predictor of one's likelihood to becoming a integrator [10]. Our work differs in that we focus on understanding of long-term active integrators who keep in handling pull requests.

Some significant works [2], [3], [4] studied factors which affected decisions to merge pull requests in GitHub. Tsay et al. found that core members both used technical and social information to evaluate potential contributions [3]. Gousios et al. investigated factors that affected the decision to merge pull requests, and factors that affected the time it took to process pull requests [2]. We also make research on pull requests, but we mainly study integrators who evaluate these pull requests.

VII. Conclusion

In this work we examine what make long-term active integrators in the evaluation of code contributions. We conduct the statistical analysis of integrator activeness in GitHub. Then we use machine learning techniques to predict long-term active integrators and discover important indicators. Results show that based on the Decision Tree, LTAPredict achieves the accuracy as 0.829, the precision as 0.81, the recall as 0.827 and the F1 as 0.818. We observe that whether integrators becoming long-term active is associated with the number of active months and social distance with contributors in their first year as integrators. These findings assist OSS projects to identify potential long-term active integrators and adopt better strategies to retain them in the evaluation of code contributions.

Acknowledgment

This work is supported by National Natural Science Foundation of China under Grant No.61300006, the State Key Laboratory of Software Development Environment under Grant No.SKLSDE-2015ZX-24, and Beijing Natural Science Foundation under Grant No.4163074.

References

[1] M. Zhou and A. Mockus, "Does the initial environment impact the future of developers?" in *Proc. of ICSE*, Honolulu, USA, May 2011.

[2] G. Gousios, M. Pinzger, and A. van Deursen, "An exploratory study of the pull-based software development model," in *Proc. of ICSE*, Hyderabad, India, July 2014.

[3] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proc. of ICSE*, Hyderabad, India, July 2014.

[4] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: The integrators perspective," in *Proc. of ICSE*, Florence, Italy, May 2015.

[5] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open source software development: What we know and what we do not know," *ACM Computing Surveys*, vol. 44, no. 2, pp. 1–35, 2012.

[6] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Leveraging transparency," *IEEE Software*, vol. 30, no. 1, pp. 37–43, 2013.

[7] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. G. Barahona, "The processes of joining in global distributed software projects," in *Proc. of GSD*, Shanghai, China, May 2006.

[8] C. Jensen and W. Scacchi, "Role migration and advancement processes in ossd projects: A comparative case study," in *Proc. of ICSE*, Minnesota, USA, May 2007.

[9] V. S. Sinha, S. Mani, and S. Sinha, "Entering the circle of trust: Developer initiation as committers in open-source projects," in *Proc. of MSR*, Honolulu, USA, May 2011.

[10] M. Gharehyazie, D. Posnett, and V. Filkov, "Social activities rival patch submission for prediction of developer initiation in oss projects," in *Proc. of ICSM*, Eindhoven, The Netherlands, September 2013.

[11] L. Dabbish, C. Stuart, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proc. of CSCW*, Washington, USA, February 2012.

[12] G. Gousios, "The ghtorent dataset and tool suite," in *Proc. of MSR*, Hyderabad, India, May 2014.

[13] T. F.Bissyande, F. Thung, D. Lo, L. Jiang, and L. Reveillere, "Popularity, interoperability, and impact of programming languages in 100,000 open source projects," in *Proc. of COMPSAC*, Kyoto, Japan, July 2013.

[14] J. Jiang, L. Zhang, and L. Li, "Understanding project dissemination on a social coding site," in *Proc. of WCRE*, Koblenz, Germany, October 2013.

[15] M. Zhou and A. Mockus, "What make long term contributors:willingness and opportunity in oss community," in *Proc. of ICSE*, Zurich, Switzerland, June 2012.

[16] E. L. Lehmann and H. J. M. D'Abrera, *Nonparametrics: Statistical Methods Based on Ranks*. Prentice-Hall, 1998.

[17] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: Activity traces and personal profiles in github," in *Proc. of CSCW*, San Antonio, USA, February 2013.

[18] X. Xia, D. Lo, X. Wang, X. Yang, S. Li, and J. Sun, "A comparative study of supervised learning algorithms for re-opened bug prediction," in *Proc. of CSMR*, Genova, Italy, March 2013.

[19] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[20] B. Shibuya and T. Tamai, "Understanding the process of participating in open source communities," in *Proc. of FLOSS*, Vancouver, Canada, May 2009.