Analysis of Github User Behavior’s

Impact on Their Contribution

Yanwu Ren Zichen Tian

1589701 1596132

[yanwu@ualberta.ca](mailto:yanwu@ualberta.ca) [ztian5@ualberta.ca](mailto:ztian5@ualberta.ca)

**ABSTRACT**

GitHub is an American company that provides [hosting](https://en.wikipedia.org/wiki/Internet_hosting_service) for [software development](https://en.wikipedia.org/wiki/Software_development) [version control](https://en.wikipedia.org/wiki/Version_control) using [Git](https://en.wikipedia.org/wiki/Git). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) functionality of Git as well as adding its own features.

It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project. GitHub offers plans for free, professional, and enterprise accounts. Free GitHub accounts are commonly used to host [open source](https://en.wikipedia.org/wiki/Open-source_software) projects. As of May 2019, GitHub reports having over 37 million users and more than 100 million [repositories](https://en.wikipedia.org/wiki/Repository_(version_control)) (including at least 28 million public repositories), making it the largest host of [source code](https://en.wikipedia.org/wiki/Source_code) in the world. In Github, users can create their own code repositories and upload the codes. Also they can interact with other users including reviewing, commenting, downloading, forking etc.

This project attempts to analyze the relationship between user's contribution to the community and their interactive activities in the platform, with emphasis on what behavior of users on the platform will lead to an increase in their contribution. To solve this problem, we must first quantify the user's interaction behaviors and contributions and use numbers to represent the user's behavior. Some machine learning and data mining methods are used to train a model to figure out the relationship between users’ contribution and interaction.

**Keywords**

Github, APIs, Machine learning, Data mining, Social Networks, Data analysis.

1. **INTRODUCTION**

The activities of Github users on the platform are mainly divided into two categories, one is uploading/updating their own code, the second is interacting with other users, including browsing, downloading, issues reporting, and staring other users' projects. This project is aimed to analyze which types of data, such as the interaction types, times, frequency and other data reflecting user activities, to observe which part of the data can have a positive impact on their contribution. For example, when a user frequently browses other users' projects for a period of time, it is most likely because the user is preparing or developing a new project, so in the future he is likely to upload the code of the new project to the platform, which will increase his contribution to the platform.

We first obtain this data. Google Bigquery is a web service launched by Google, which allows developers to use Google's architecture to run SQL statements to operate on super large databases. Bigquery allows users to upload their super large amount of data and conduct interactive analysis directly through it, so that they do not need to invest in building their own data center. In this project, we download the githubarchive datasets from the Google BigQuery website ([https://bigquery.cloud.google.com](https://bigquery.cloud.google.com/)). All of the datasets are from January 1st 2018 to December 31st 2018 (12months). All of the datasets are public repositories and in total there are 475,516,715 events and 14 categories.

The interaction behavior can be quantified as the amount of code viewed by a user, the number of reported issues, and the number of stars giving, etc.; the contribution can be quantified as the amount of code uploaded/updated, the number of times the code is viewed, and so on.

Then we use XGBoost, a machine learning model to train the datasets and make prediction of relations between the reputation and the user’s activities.

1. **DATASETS**

Due to the visit rate limitation of Github API, if there is no authorization for direct access, the single IP limit is 60 requests per hour. 5000 per hour if authorized. Bigquery reduces some of the implementation challenges of analyzing large datasets. Price models facilitate the discovery of statistical analysis skills, so useful information can be derived from data samples rather than brute force analysis for all data sets. Bigquery is designed to analyze billions of rows of similar data, using SQL like syntax. It is not a substitute for SQL database and is not suitable for transaction processing applications. Bigquery supports analysis interaction style. So we use Google BigQuery to download the big datasets, which contains 475,516,715 events, 14 categories of public repositories during 12 months (January 1st 2018 to December 31st 2018). Here is a dataset example shown in Table 1.

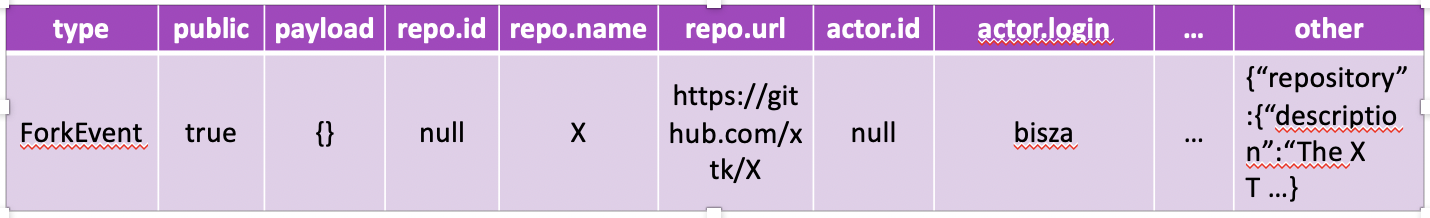


Table 1. An example of the dataset.

We can see from Table 1 that a dataset contains many information such as: event type, public or not, repository link, actor ID etc. In this project we only use event type and actor ID.

In the datasets, there are fourteen different event types. PushEvent: Triggered on a push to a repository branch. Branch pushes and repository tag pushes also trigger webhook push events. WatchEvent: Triggered when someone stars a repository. This event is not related to watching a repository. See this API blog post for an explanation. CreateEvent: Represents a created branch or tag. CommitCommentEvent: Triggered when a commit comment is created. DeleteEvent: Represents a deleted branch or tag. GollumEvent: Triggered when a Wiki page is created or updated. ForkEvent: Triggered when a user forks a repository. IssueCommentEvent: Triggered when an issue comment is created, edited or deleted. IssuesEvent: Triggered when an issue is opened, edited, deleted, pinned, unpinned, closed, reopened, assigned, unassigned, labeled, unlabeled, locked, unlocked, transferred, milestoned, or demilestoned. MemberEvent: Triggered when a user accepts an invitation or is removed as a collaborator to a repository, or has their permissions changed. PublicEvent: Triggered when a private repository is made public. PullRequestEvent: Triggered when a pull request is assigned, unassigned, labeled, unlabeled, opened, edited, closed, reopened, synchronize ready for review, locked, unlocked or when a pull request review is requested or removed. PullRequestReviewCommentEvent: Triggered when a comment on a pull request's unified diff is created, edited, or deleted. ReleaseEvent: Triggered when a release is published, unpublished, created, edited, deleted, or prereleased.

Figure 1 shows the distribution of the 14 categories of events. In the figure, PushEvent is of the largest amount (much higher than all the other events), CommitCommentEvent is of the smallest amount.

手机屏幕截图

描述已自动生成

Figure 1. Event distribution.

1. **METHODOLOGY**

In this section we will talk about the methodology of our project, including how we quantify the interaction and reputation of the user activities in Github, and how we process the data, and how we use the data to train a machine learning model.

* 1. **Quantification**

First we define a discipline to quantify how the user activities will influence the interaction and reputation of a Github user.

Here follows the two desciplines:

(1)

And

(2)

* 1. **Data Process**

Get the number of events for every users in each month as a dataset. So we get 12 datasets for 12 months.

地图的截图

描述已自动生成

Figure 2. Data trend of 12 months

Figure 2 shows the activity and events evolution trend of 12 months.

Pearson correlation coefficient:

(3)

cov is the covariance; is the standard deviation of X; is the standard deviation of Y.

图片包含 游戏机

描述已自动生成

Figure 3. Pearson correlation coefficient for 14 event categories

Figure 3 shows the Pearson correlation coefficient for 14 event categories.

Concatenate the 12 datasets together. Take the contribution value below 98% quantile (maximum value: 40,000, mid: 5). Interaction cannot be directly used to predict users’ contributions. We need to use the separate interaction data as the features. Then we get the final dataset, featuring: Watch, IssueComments, Issues, Member, PullRequest and Fork. Table 2 shows an example of the final dataset.

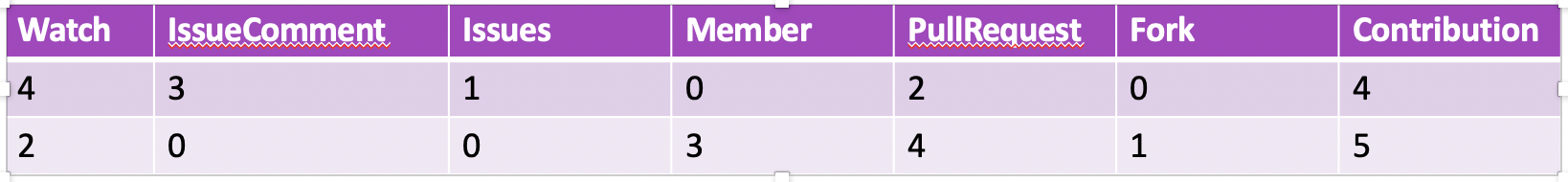


Table 2. An example of the final dataset.

* 1. **Machine Learning Model**

XGBoost is a machine learning function library focusing on gradient lifting algorithm, which was invented in February 2014. This function library has been widely concerned because of its excellent learning effect and efficient training speed. XGBoost not only has a good learning effect, but also has a fast speed. Compared with the implementation of gradient lifting algorithm in another commonly used machine learning library scikit-learn, the performance of XGBoost is often improved by more than 10 times.

The idea of the algorithm is to add trees, split the features to grow a tree, and add a tree at a time. In fact, it is to learn a new function to fit the residual predicted last time. When we get k trees after training, we need to predict the score of a sample. In fact, according to the characteristics of the sample, a corresponding leaf node will fall in each tree, and each leaf node corresponds to a score. Finally, we only need to add up the corresponding score of each tree to be the predicted value of the sample.

(4)

(5)

Where is the fraction of leaf node Q, and is one of the regression trees.

XGBoost objective function is defined as:

(6)

The objective function consists of two parts, the first part is used to measure the difference between the predicted score and the real score, and the other part is the regularization term.

(7)

Where i represents the i-th sample, represents the prediction error of the i-th sample. The smaller the error, the better. The following represents the function of the complexity of the tree. The smaller the complexity, the lower the generalization ability.

(8)

The regularization term also includes two parts, T is the number of leaf nodes, and is the score of the leaf nodes. γ can control the number of leaf nodes, and λ can control the score of leaf nodes not to be too large to prevent overfitting. The newly generated tree is to fit the residuals of the last prediction, that is, when t trees are generated, the prediction score can be written as:

(9)

XGBoost idea is to use its Taylor second order at to approximate it. The prediction score of the first t-1 trees and the residuals of y have no effect on the optimization of the objective function, and can be removed directly. At the same time, the objective function can be rewritten as:

(10)

Where is the first derivative and is the second derivative. Rewrite the objective function as a univariate quadratic function about the leaf node score w, and it becomes very simple to solve the optimal value of w and the objective function, just use the vertex formula directly. Therefore, the optimal w and objective function formula is:

(11)

(12)

XGBoost is widely used in data competitions and industry because of its many advantages:

1. Use many strategies to prevent overfitting, such as: regularization terms, Shrinkage and Column Subsampling, etc.
2. Support parallelization. Although the tree is a serial relationship, the nodes at the same level can be parallel. Specifically, for a certain node, the optimal split point is selected within the node, and the candidate split point calculation gain is multi-threaded in parallel. This makes training very fast.
3. Added processing for sparse data.
4. Cross-validation, early stop, when the prediction result is already good, you can stop building in advance to speed up the training speed.
5. Support setting sample weights. The weights are reflected in the first derivative *g* and the second derivative *h*. By adjusting the weights, you can pay more attention to some samples.

In this project, we split the datasets into two parts, 80% of the datasets are used to train the model, 20% are used to test (Figure 4). Use 5-fold cross-validation on train set to tuning the model.

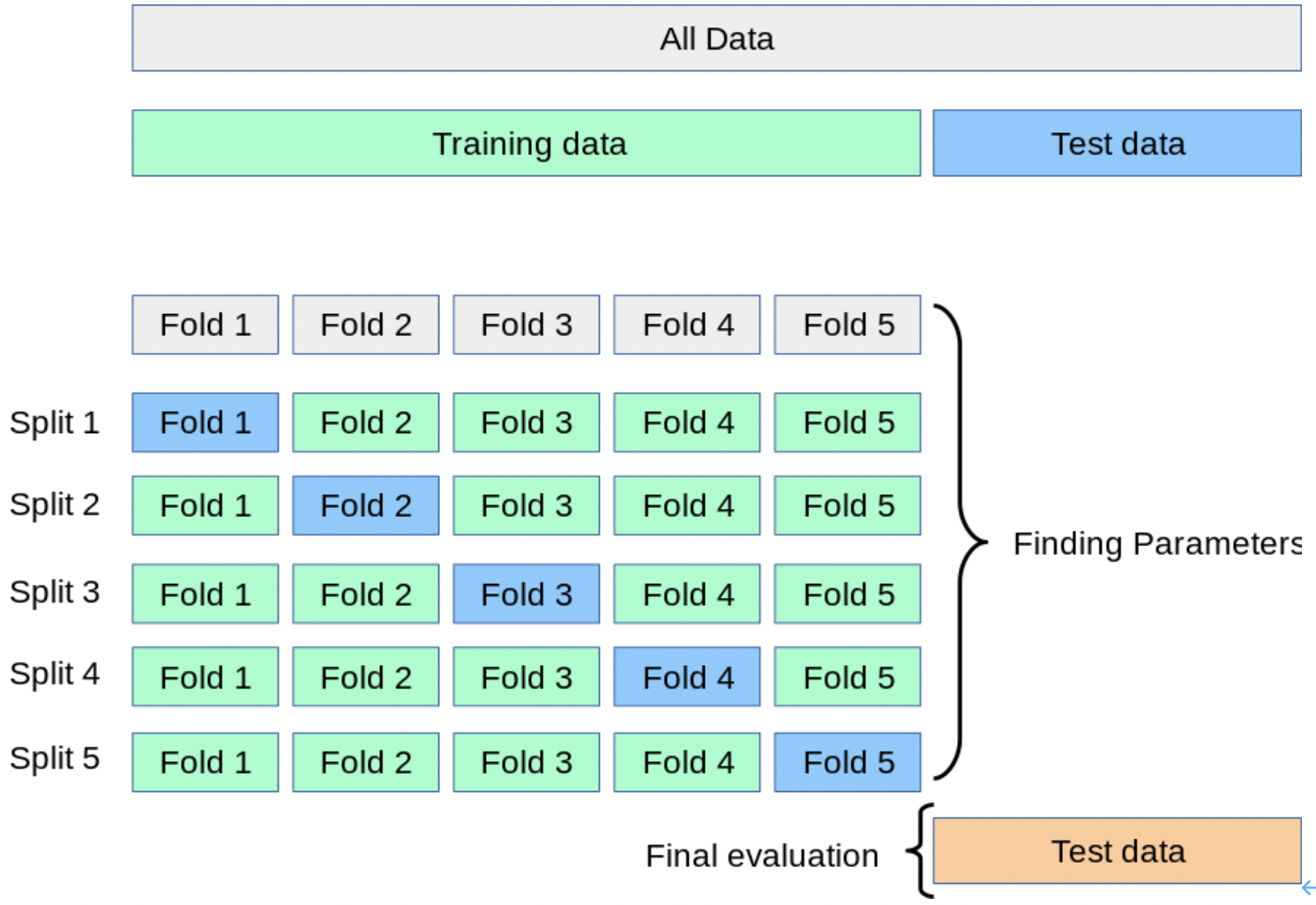


Figure 4. Datasets split

1. **RESULTS**

We test mean absolute error and mean absolute percentage error for both the whole test sets and 95% test data.

(13)

(14)

For the whole test sets: MAE = 0.897, MAPE = 17.9%. For 95% test data: MAE = 0.0519, MAPE = 1.03%. The other 5% test data have a big difference between the true label and the predicted value. Figure 5 shows the importance of the 6 kinds of events.

1. **CONCLUSION**

For most users, we can predict their contribution according to their interaction behaviors. For a small

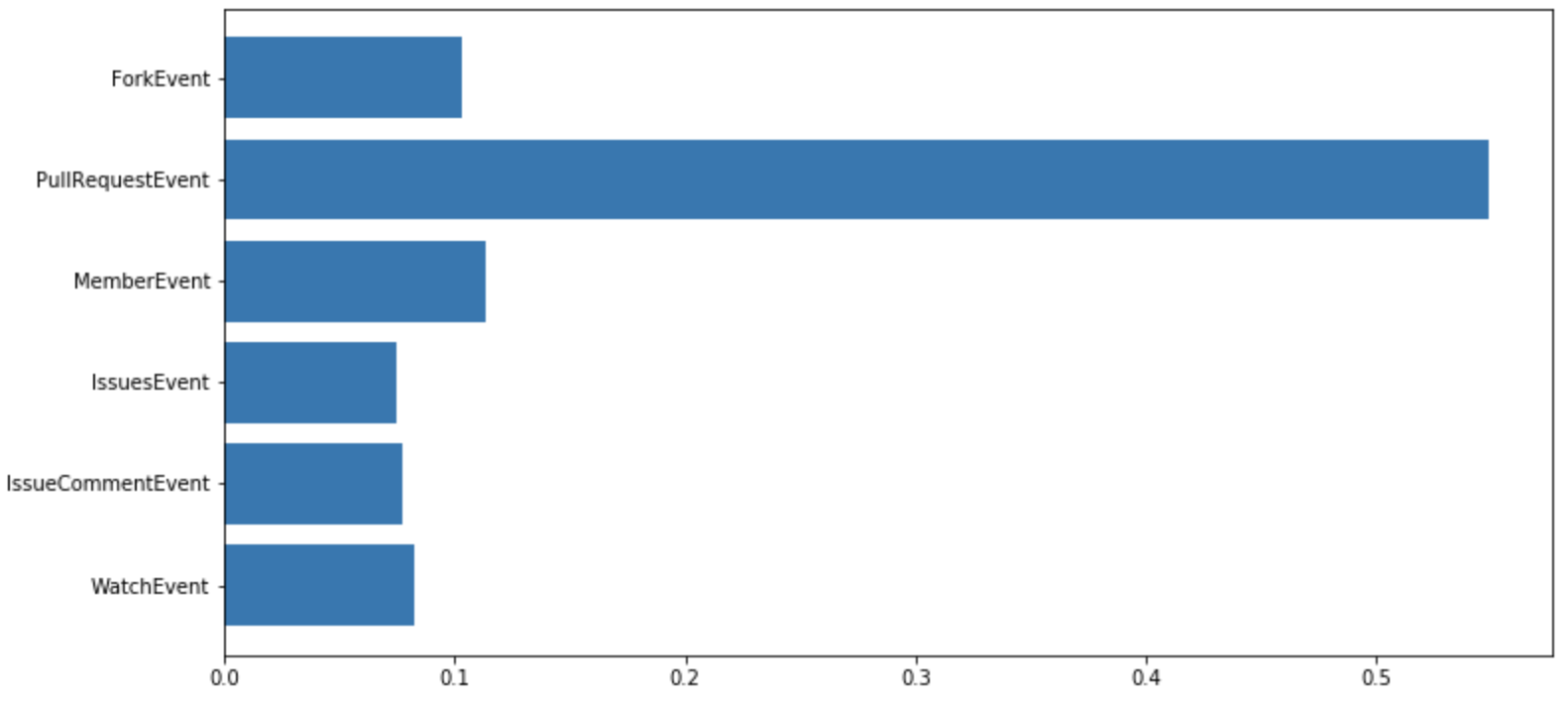


Figure 5. Importance of six categories of events

number of users, Their contribution behaviors and interaction behaviors do not conform to the rules given by the model. For future works we can add different weights for events in interaction and contribution definitions. Make model prediction more accurate and analyze more user behavior rules based on user event data

1. **REFERENCE**