

Identifying Unmaintained Projects in GitHub

メンテナンスされていない GitHubプロジェクトの識別

著者

Jailton Coelho, Marco Tulio Valente, Luciana L. Silva
(Federal University of Minas Gerais, Brazil)
Emad Shihab (Concordia University, Canada)

論文情報：

ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
(ESEM '18), October 11-12, 2018, Oulu, Finland. ACM, New York, NY, USA, 10 pages.

発表者：堀口 日向

1. 背景

OSSプロジェクトが継続的にメンテナンスされているかどうか

⇒ そのOSSを使うか判断するのに重要

GitHubには
プロジェクトがメンテナンスされているかどうかを
明確に判断するための機能がない

⇒ コミット履歴やIssue, PRの状態を総合的に見て
ユーザ自身が判断する必要

2. 既存研究

既存研究[8][14][15][24]で使われた
メンテナンスされているかどうかの判断基準

過去1年間にコミットがあるかどうか

⇒ 1年間待たないと分からない？

プロジェクトの様々な開発活動の指標（コミット数やPR数など）と
機械学習モデルを使えば
1年待たずとも判断できるのでは？

3. 目的

ランダムフォレストを用いた機械学習で
プロジェクトがメンテナンスされているかどうかを
判断するモデルを作成

モデルが実際に利用できるか検証

- RQ1. 実際にモデルを使用した際のAccuracyはどれぐらいか？
- RQ2. 実際にモデルを使用した際のRecallはどれぐらいか？
- RQ3. メンテナンスしていないと判断するのに
コミットされなくなってから1年かかるか？

4. 機械学習モデルの作成 | データセット (1/2)

1. 学習用データの収集

GitHubで最もスターの多いプロジェクトの上位10,000件を取得
以下のプロジェクトは除外

- 最初のコミットから最後のコミットまで2年未満
- プログラミング言語で書かれたファイルがない (HTML, CSS等)
- ソフトウェアではない (ドキュメント, awesome-lists)

⇒ 6,785件を抽出

4. 機械学習モデルの作成 | データセット (2/2)

2. メンテナンスされていないプロジェクトの識別

- 先月（データ収集当時から）に少なくとも1つのリリース
メンテナンスされていると判断
⇒ 754件
- メンテナンスされていないと明示的に宣言＋アーカイブ化
(GitHubの機能. リポジトリがReadOnlyになる)
⇒ 248件

4. 機械学習モデルの作成 | 入力する特徴量 (1/3)

Table 1: Features used to identify unmaintained projects.

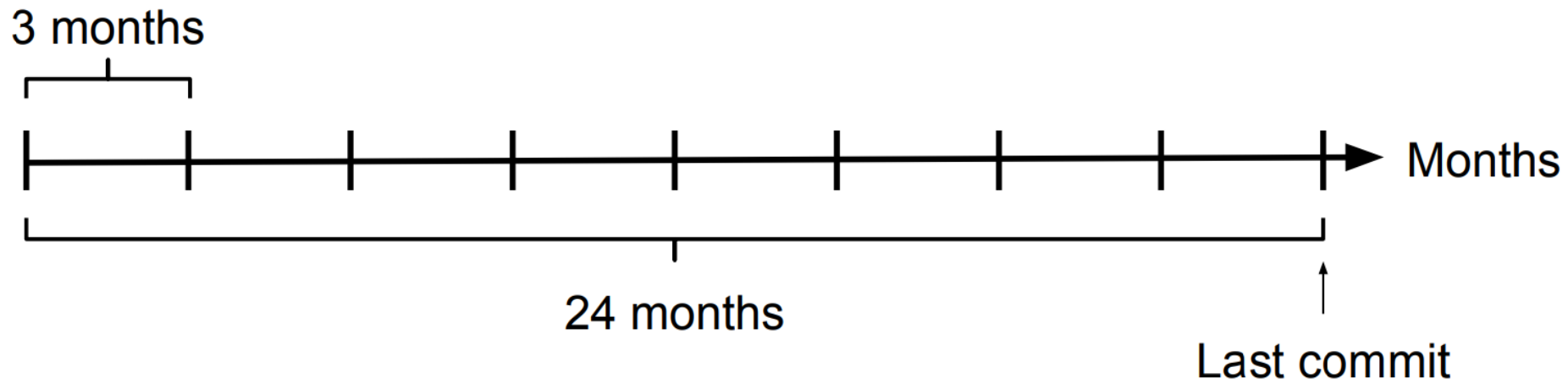
Dimension	Feature	Description
Project	Forks	Number of forks created by developers
	Open issues	Number of issues opened by developers
	Closed issues	Number of issues closed by developers
	Open pull requests	Number of pull requests opened by the project developers
	Closed pull requests	Number of pull requests closed by the project developers
	Merged pull requests	Number of pull requests merged by the project developers
	Commits	Number of commits performed by developers
	Max days without commits	Maximum number of consecutive days without commits
	Max contributions by developer	Number of commits of the developer with the highest number of commits
Contributor	New contributors	Number of contributors who made their first commit in the considered period
	Distinct contributors	Number of distinct contributors that committed in the considered period
Owner	Projects created by the owner	Number of projects created by a given owner
	Number of commits of the owner	Number of commits performed by a given owner

4. 機械学習モデルの作成 | 入力する特徴量 (2/3)

特徴量の算出方法

- ・プロジェクトの最後のコミットから過去nか月を対象
- ・各特徴量をmか月間隔で取得

例：nが24か月，mが3か月 $\Rightarrow$ 8つのデータポイント



4. 機械学習モデルの作成 | 入力する特徴量 (3/3)

nとmのさまざまな組み合わせを計10シナリオ用意

過去nか月分

Scenario	1	2	3	4	5	6	7	8	9	10
Length	6	6	12	12	12	18	18	24	24	24
Intervals	3	6	3	6	12	3	6	3	6	12
Data points	26	13	52	26	13	78	39	104	52	26

mか月間隔

4. 機械学習モデルの作成 | 学習

計10シナリオのデータセットでランダムフォレストを学習

ランダムフォレストを採用した理由

- ・ノイズや外れ値に対する堅牢性[36]
- ・他の多くのソフトウェアエンジニアリングの研究で採用

⇒ 5分割交差検証を100ラウンド実施し，評価指標を平均

4. 機械学習モデルの作成 | 学習結果

Table 4: Prediction results (mean of 100 iterations, using 5-cross validation); best results are in bold.

Metrics	0.5 Year		1 Year			1.5 Years		3 months	2 Years	
	3 months	6 months	3 months	6 months	12 months	3 months	6 months		6 months	12 months
Accuracy	0.90	0.91	0.91	0.90	0.89	0.91	0.90	0.92	0.91	0.90
Precision	0.83	0.87	0.87	0.84	0.82	0.86	0.83	0.86	0.85	0.83
Recall	0.78	0.74	0.77	0.75	0.72	0.78	0.76	0.81	0.79	0.73
F-measure	0.80	0.79	0.81	0.79	0.77	0.82	0.79	0.83	0.82	0.78
Kappa	0.74	0.74	0.76	0.73	0.70	0.76	0.73	0.78	0.76	0.71
AUC	0.86	0.85	0.86	0.85	0.83	0.87	0.85	0.88	0.87	0.84

各評価指標が最も高いモデル

4. 機械学習モデルの作成 | 学習結果

最良モデルとベースラインとの比較

Baseline #1 : すべてのプロジェクトがメンテナンスされていないと予測
Baseline #2 : ランダムな予測

Metrics	Model	Baseline #1	Baseline #2
Accuracy	0.92	0.22	0.49
Precision	0.86	0.22	0.22
Recall	0.81	1.00	0.48
F-measure	0.83	0.37	0.30
Kappa	0.78	0.00	0.01
AUC	0.88	0.50	0.49

最良モデルの評価指標の方が高い

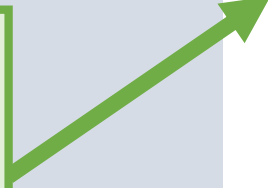
4. 機械学習モデルの作成 | 重要性の高い特徴量

ランダムフォレストの学習に利用した特徴量のうち
重要度 (MDA) の上位5つをしてみる

22~24か月目の
コミット数



22~24か月目の
コミットがない
最大日数



Feature	Period	MDA
Commits	T _{22,24}	38.5
Max days without commits	T _{22,24}	28.6
Max days without commits	T _{10,12}	21.9
Max contributions by developer	T _{16,18}	21.1
Closed issues	T _{1,3}	18.0

5. モデルが実際に使えるか？ | Accuracy (RQ1)

方法

1. 学習に利用していないプロジェクト5,783件に最良モデルを適用しメンテナンスされていないと判断したプロジェクトを取得
2. プロジェクトオーナーのメアドが公開されている323件を抽出し実際にメンテナンスしていないか聞く
3. 実際にメンテナンスしていなかった割合をAccuracyとする

5. モデルが実際に使えるか？ | Accuracy (RQ1)

返答をカテゴリ化（返答率は37.1%）

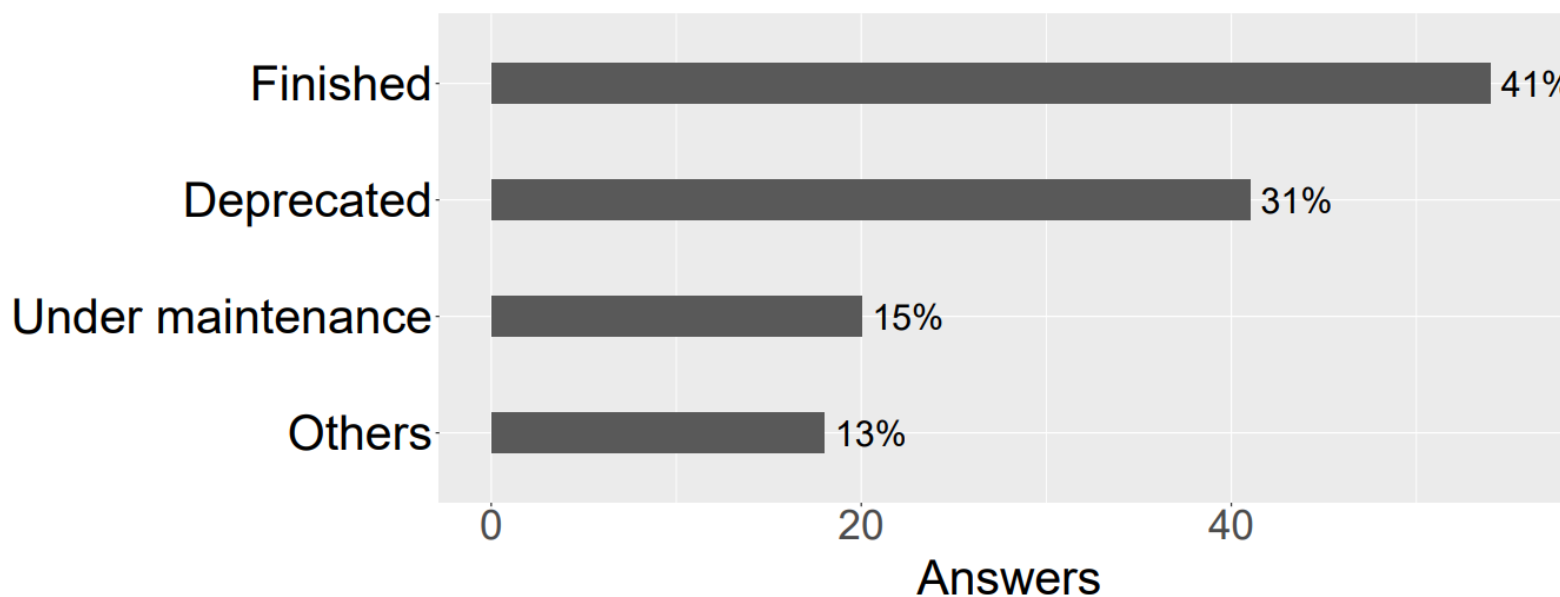


Figure 3: Survey answers about projects' status.

- 完成 : 41%
- 非推奨 : 31%
- その他 : 13%のうち,
8%はプロジェクトが
停止していると回答



80%のプロジェクトが
メンテナンスしていない

Accuracy : 80%

5. モデルが実際に使えるか？ | Recall (RQ2)

方法

1. 学習に利用していないプロジェクト5,783件から，以下のようなメンテナンスされていないことを意味する文章[8]を含んでいるREADMEがあるプロジェクトを検索

*no longer under development, no longer supported or updated,
deprecation notice, dead project, deprecated, unmaintained,
no longer being actively maintained, not maintained anymore,
not under active development, no longer supported,
is not supported, is not more supported, no longer supported,
no new features should be expected, isn't maintained anymore*

訳：
もはや
開発されて
いない

⇒ 451件 (7.8%) のプロジェクトを抽出

5. モデルが実際に使えるか？ | Recall (RQ2)

方法

2. 451件のプロジェクトを目視
⇒ 112件（25%）が実際にメンテナンスされていないことを確認
3. これら112件のプロジェクトに最良モデルを適用
メンテナンスされていないと判断した割合をRecallとする

⇒ 108件をメンテナンスされていないと判断

Recall : 96%

5. モデルが実際に使えるか？ | 1年以内にわかるか？ (RQ3)

RQ1では、103件のプロジェクトは
メンテナンスされていないとモデルが判断

このうち77件を調べると
過去1年間にコミットがある

⇒ 最後のコミットから1年間待たずとも
メンテナンスされていないと判断することができた

6. 結論

メンテナンスされていないGitHubプロジェクトを識別する 機械学習モデルを提案

- 実際のプロジェクトで検証し，80%のAccuracy，96%のRecall
⇒ 実際利用できるほど精度が高い
- 既存研究では，メンテナンスされていないと判断するのに
最後のコミットから1年間待つ必要
⇒ 提案モデルでは，最後のコミットから1年待たなくても
メンテナンスされていないプロジェクトを識別可能