Journal of KIIT. Vol. 20, No. 12, pp. 167-174, Dec. 31, 2022. pISSN 1598-8619. eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2022.20.12.167

기계학습을 이용한 오픈소스 생존 가능성 예측

박소희*[1], 권령구*[2], 권기현**

Prediction of Survivability for Open-Source with Supervised Machine Learning

Sohee Park*[1], Ryeonggu Kwon*[2], and Gihwon Kwon**

이 논문은 과학기술정보통신부 및 정보통신기술진흥센터의
2022년 SW공학기술 역량강화 지원사업의 연구결과로 수행되었음(S0436-22-1007)

요 약

현대의 소프트웨어 개발 프로젝트에서는 다양한 오픈소스가 사용되고 있다. 그러나 많은 오픈소스들 중 지원이 중단되는 오픈 소프트웨어들도 존재한다. 오픈소스 지원이 중단된다면 언어 노후화 및 오류 미해결 등의 문제가 발생해 개발 프로젝트에 심각한 영향을 끼친다. 이러한 문제를 최소화하기 위해서 생존 가능성이 높은 오픈소스를 사용해야 한다. 본 논문에서는 기계학습을 이용한 오픈소스 프로젝트의 생존 가능성 예측 방법을 제안한다. 여러 오픈소스들의 프로젝트 개발 추세, 버그 이슈 해결 추세, 평균 기여도 추세를 예측하고, 이를 바탕으로 오픈소스의 생존 가능성을 성장기, 정체기, 쇠퇴기 세 유형으로 분류하였다. 제안한 방법을 통해 생존 가능성이 높은 프로젝트는 개발, 버그, 기여도 3가지 모두 증가하는 반면, 생존 가능성이 낮은 프로젝트는 개발량이 적거나 3가지 모두 감소하는 것을 알 수 있었다.

Abstract

Open-sources are used in modern software development projects. However, some of open-sources are interrupted. Discontinued support of open-sources causes problems such as aging of language and unresolved errors, which seriously affects development projects. To minimize these problems, open-sources with high survivability should be used in software development. In this paper, we propose a method for predicting the survivability of open-sources using machine learning. We predicted development, bug issue resolution, and average contribution tendencies, and categorized survivability of open-sources as Growth, Stagnation, and Decline. Through the proposed method, we found that projects with high survivability tend to increase all in development, bugs, and contributions, while projects with low survivability either have less development or decrease all.

Keywords
open-source project, survivability, prediction, machine learning

* 경기대학교 AI컴퓨터공학부
- ORCID[1]: https://orcid.org/0000-0003-0530-4946
- ORCID[2]: https://orcid.org/0000-0002-4942-247X
** 경기대학교 AI컴퓨터공학부 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-8221-4939

· Received: Nov. 02, 2022, Revised: Dec. 19, 2022, Accepted: Dec. 22, 2022
· Corresponding Author: Gihwon Kwon
Dept. of Computer Engineering, Kyonggi University, 154-42, Gwanggyosan-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Korea
Tel.: +82-31-249-9666, Email: khkwon@kgu.ac.kr

Ⅰ. 서 론

깃허브는 온라인 분산 버전 관리 저장소 서비스로 많은 오픈소스들이 깃허브를 통해 공유되고있다[1]. 오픈소스는 개발 편의성을 향상시켜주는 다양한 프로그램 및 라이브러리 등을 제공한다[2]. 이러한 점 때문에 현대의 소프트웨어 개발 프로젝트에서는 다양한 오픈소스가 사용되고 있다[3]. 하지만 개발이 중단되는 오픈소스들도 존재한다. 개발 프로젝트에서 오픈소스가 많이 사용되는 만큼 오픈소스의 개발 중단은 프로젝트의 유지보수성, 안정성에 큰 영향을 미친다. 이러한 문제를 예방하기 위해서는 미래에도 지속적으로 기능 개발 및 버그 수정이 이루어지는 오픈소스 즉, 생존 가능성이 높은 오픈소스를 사용해야 한다.

오픈소스의 생존 가능성을 식별하기 위해서 기존 연구들은 개발에 참여하는 기여자의 장기 참여 여부와 프로젝트 코드의 복잡도 등을 고려하였다[4][5]. [4]에서는 오픈소스의 장기 기여자가 프로젝트의 생존 가능성에 중요한 요소로 보고 기여자에 초점을 맞추었다. [5]에서는 제품, 유저, 개발자 3가지 요소를 종합하여 오픈소스의 생존 가능성을 예측하였다. 특히, 제품과 코드의 복잡도를 고려하였다. 제품, 코드의 복잡도가 높을수록 사람들의 이해도가 떨어져 생존 가능성이 낮아진다고 밝혔다.

본 논문에서는 지도학습 기법인 회귀 분석을 이용하여 오픈소스의 생존 가능성을 예측한다. 생존 가능성을 식별하기 위해서 개발량, 버그 이슈 생성 및 해결, 기여도를 예측하였다. 소프트웨어의 유지보수, 안정성, 지속성을 위해선 지속적인 개발 활동과 버그 발견 및 해결이 필요하다. 또한, 오픈소스 프로젝트는 기여자들의 참여로 진행된다. 따라서 본 논문에서는 오픈소스 생존의 핵심을 개발과 유지보수의 지속 여부로 판단하였기 때문에 개발량, 버그 발견 및 해결, 기여도를 예측에 사용하였다.

제안한 방법으로 예측한 결과 개발, 버그 이슈, 기여도에 따라 오픈소스를 성장기, 정체기, 쇠퇴기로 분류하였다. 유형에 따라 생존 가능성을 추정할 수 있었으며, 이를 통해 생존 가능성이 높은 오픈소스의 특징을 구체화하였다.

논문의 구성은 다음과 같다. 2장에서 배경지식을 설명하고, 3장에서는 생존 가능성 예측을 제안한다. 4장에선 예측 결과 및 분석 결과를 보이고, 마지막 5장에서는 결론 및 향후 연구에 대해 보인다.

Ⅱ. 배경지식

2.1 기계학습

기계학습이란, 학습데이터로부터 특징(Feature)을 추출하고, 특징을 학습하여 클래스를 구분짓는 모델이나 상관관계를 예측하는 모델을 생성한 후, 새로운 데이터를 통해 원하는 값(Target)을 추론하는 과정을 수행한다[6]. 기계학습이 주로 이용되는 문제에는 패턴 인식, 회귀, 함수 근사화 등이 있다. 이러한 유형의 문제를 해결하기 위한 기계학습 학습 알고리즘은 크게 지도학습, 비지도학습, 강화학습으로 나뉘어진다.

지도학습은 입력 데이터와 기대 출력 데이터를 이용하여 학습하는 학습 알고리즘이다[7]. 데이터가 입력되었을 때, 출력 데이터가 최대한 기대 출력 데이터에 가깝게 되도록 학습한다. 주로 회귀 또는 분류 문제에 사용된다. 그에 반해, 비지도학습은 입력 데이터만을 가지고 학습하는 알고리즘이다. 입력 데이터를 분석하여 특징, 의미를 찾고, 분석하는 것을 목적으로 한다. 군집화 문제나 자기 조직화 지도 문제에서 자주 사용된다. 마지막으로 강화학습은 입력 데이터의 특징을 토대로 행동을 결정하는 알고리즘이다. 행동을 실행한 후 변화한 환경에 따라 보상 값이 주어지고, 이 보상 값이 최대가 되는 방향으로 학습한다. 강화학습은 동적 환경을 다루는 게임 이론 및 제어 이론 문제 등에 사용된다.

본 논문에서는 지도, 비지도, 강화 학습 중 지도학습의 회귀 분석을 사용하여 생존 가능성을 예측하였다. 회귀 분석은 변수들간의 상관관계를 분석하여 예측 모델을 형성하는 방법이다. 크게 선형 분석과 비선형 분석 방법으로 나눠진다. 선형 분석은 독립 변수와 종속 변수의 상관관계를 선형으로 모델링하는 방법이고, 비선형 분석은 선형 분석과 다르게 두 변수간의 상관관계를 곡선, 비선형으로 모델링하는 방법이다[8].

오픈소스 생존 가능성을 나타내는 상관관계가 비선형적어서 본 논문에서는 비선형 회귀 분석을 이용하였다[9].

2.2 비선형 회귀 분석

비선형 회귀 분석은 선형 회귀 분석과 다르게 독립 변수 x와 종속 변수 y간의 상관관계를 곡선으로 모델링하는 분석 방법이다[9]. 비선형 회귀 분석 중 하나인 다항 회귀 분석은 두 변수간의 상관관계에 대한 n차 다항 모델을 형성하는 기법이다[10]. 다항 회귀 기법은 식 (1)과 같이 독립 변수 x와 종속 변수 y간의 관계에 가장 적합한 선을 설명하는 n차 방정식을 사용한다.

$$y = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon \qquad (1)$$

식 1의 독립 변수 x는 다른 변수의 값에 영향을 받지 않는 변수로 원인에 해당하는 값을 독립 변수로 설정한다. 기계학습에선 feature로 사용한다. 종속 변수 y는 독립 변수 값에 영향을 받는 변수로 결과에 해당하는 변수를 종속 변수로 설정한다. 기계학습에서는 target을 종속 변수로 사용한다. β_i $(i = 1, 2, \cdots, n)$는 회귀 계수로 각 x_i에 대한 기울기를 나타낸다. ϵ는 오류 값으로 실제 값과 예측 값 간의 오차를 나타낸다.

다항 회귀 기법은 두 변수들간의 관계를 분석하고, 이를 이용해 미래의 값 예측이 필요한 여러 분야에서 사용되고 있다.

III. 생존 가능성 예측

3.1 데이터 구성

본 논문에서는 깃허브의 데이터를 이용하여 오픈소스의 생존 가능성을 예측하였다. 개발, 버그 생성 및 해결, 기여도 추세를 예측하기 위해 사용한 데이터는 표 1과 같다.

날짜 데이터는 커밋, 풀 리퀘스트, 이슈 등이 생성되거나 변경된 날짜이다.

표 1. 예측에 사용된 데이터 구성
Table 1. Data set used for prediction

Category	Data type	Role
Common	date	feature data
Development	commit	feature / target data
	pull request	target data
Bug Issue	all bug issue	feature / target data
	closed bug issue	target data
Contribution	contributors	-
	commit of contributors	
	pull request of contributors	
	C_{avg}	target data

날짜 데이터는 개발, 버그, 기여 예측에 공통으로 사용하였다. 3가지 예측 모두 시간의 경과에 따른 예측이 필요하고, 특정 날짜의 데이터를 예측하기 때문에 날짜 데이터를 feature 데이터로 설정하였다.

커밋과 풀 리퀘스트는 개발 추세 예측에 사용하였다. 커밋은 깃의 기본 저장 단위로, 변경이력이 저장되어 커밋으로 개발 진행 여부를 알 수 있다. 또한, 여러 커밋이 생성된 후 풀 리퀘스트가 수행된다. 따라서 커밋을 풀 리퀘스트 예측에 사용할 feature 데이터로 선정하였다. 하지만 미래의 풀 리퀘스트를 예측하기 위해선 미래의 커밋 수 또한 예측해야 한다. 따라서 커밋을 feature / target 데이터로 설정하였다. 풀 리퀘스트는 배포 버전에 개발 사항을 적용하기 위해서 수행된다. 따라서 이 데이터를 통해 배포 버전의 변화를 알 수 있기 때문에 개발 추세 예측의 target 데이터로 사용하였다.

전체 버그 이슈 개수, 해결된 버그 이슈 개수는 버그 해결 추세 예측에 사용하였다. 사용자가 버그를 발견하는 경우 버그 이슈를 작성한다. 그리고 개발자가 생성된 버그 이슈를 해결한 경우 이슈 상태를 closed로 변경한다. 전체 버그 이슈 개수로 지금까지 발견된 버그 수를 알 수 있고, 해결된 버그 이슈 개수로는 지금까지 해결된 버그 수를 알 수 있다. 그래서 버그 생성 및 해결 추세 예측에 두 데이터를 사용하였다.

기여도 예측에는 기여자 수, 기여자들의 커밋, 풀 리퀘스트 수를 이용하였다. 기여자들의 오픈소스 기

여도를 계산하기 위해 커밋과 풀 리퀘스트를 사용하였다. 이 두 데이터를 합쳐 식 (2)과 같이 기여도 C로 환산하였다.

$$C = \sqrt{commit^2 + pull\ request^2} \quad (2)$$

또한, 각 기여자들의 기여도를 하나의 데이터로 병합하기 위해 식 (3)과 같이 평균 기여도 C_{avg}를 계산하였다. 이 C_{avg}를 기여자들 전체의 기여도를 나타내는 데 사용하였다.

$$C_{avg} = \frac{\sum_{i=1}^{n} C_i}{number\ of\ contributors} \quad (3)$$

3.2 예측 방법

개발 / 버그 이슈는 다섯 단계, 기여도의 경우는 세 단계에 걸쳐서 예측을 수행하였다. 개발, 버그 이슈 추세 예측의 경우 각각 커밋 수, 전체 버그 이슈 수를 추정하고, 추정된 데이터를 바탕으로 풀 리퀘스트 수, 해결된 버그 이슈 수를 예측하였다. 예측은 그림 1과 같이 진행하였다. 예측 수행 단계는 다음과 같다.

- 날짜 데이터와 feature / target 데이터와 target 데이터를 수집한다.
- 수집된 데이터를 이용하여 다항 회귀 분석을 통해 예측 모델을 형성한다.
- 모델에 예측하고 싶은 날짜들을 입력하여 feature / target 데이터를 예측 및 생성한다.
- 날짜 데이터와 feature / target 데이터, target 데이터로 예측 모델을 형성한다.
- 예측하고 싶은 날짜 데이터와 단계 3에서 생성한 데이터를 모델에 입력하여 target 데이터를 예측 및 생성한다.

IV. 사례 적용 및 분석

4.1 오픈소스 선택

본 논문에서는 식 4, 5, 6을 적용하여 예측에 사용할 오픈소스를 선택하였다. 식 4, 5의 $Commit_{avg}$는 최근 한 달 이내로 범위를 설정하였다.

$$Commit_{avg} = \frac{number\ of\ commits}{1\ week} \geq 50 \quad (4)$$

$$Commit_{avg} < 7 \quad (5)$$

$$Popularity = number\ of\ stars \geq 50{,}000 \quad (6)$$

$Commit_{avg}$는 개발량의 기준, $Popularity$는 인기도의 기준으로 사용하였다. $Commit_{avg}$가 50 이상이면 개발량이 많은 오픈소스, 7 미만이면 하루 평균 1개의 커밋도 수행되지 않았기에 개발량이 적은 오픈소스로 보았다. $Popularity$는 오픈소스의 star 수로[11], 50,000 이상이면 인기도가 많은 오픈소스로 보았다. 식 4, 5, 6을 바탕으로 총 6개의 오픈소스를 선택하였다. 선택한 오픈소스는 표 2와 같다.

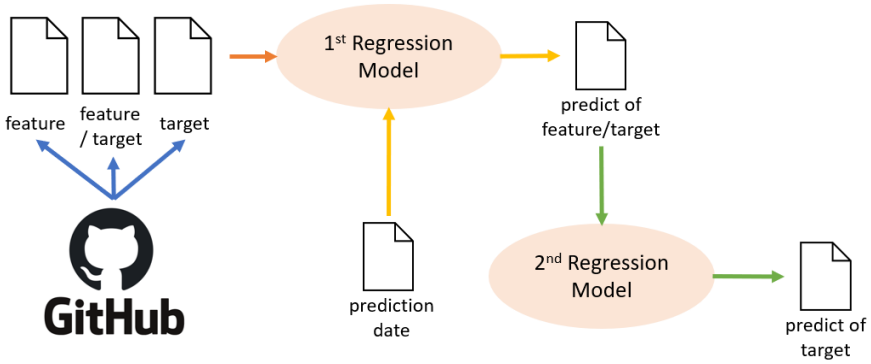

그림 1. 생존 가능성 예측 과정
Fig 1. Survivability prediction process

표 2. 오픈소스 정보
Table 2. Data of open-sources

Open-source	Popularity	Average of commits
Mastodon	37,300	84
Terminal	86,300	7.7
Drawio	32,300	1.2
Angular	85,300	59.9
React	199,000	13.5
Vue	201,000	1.9

4.2 예측 결과

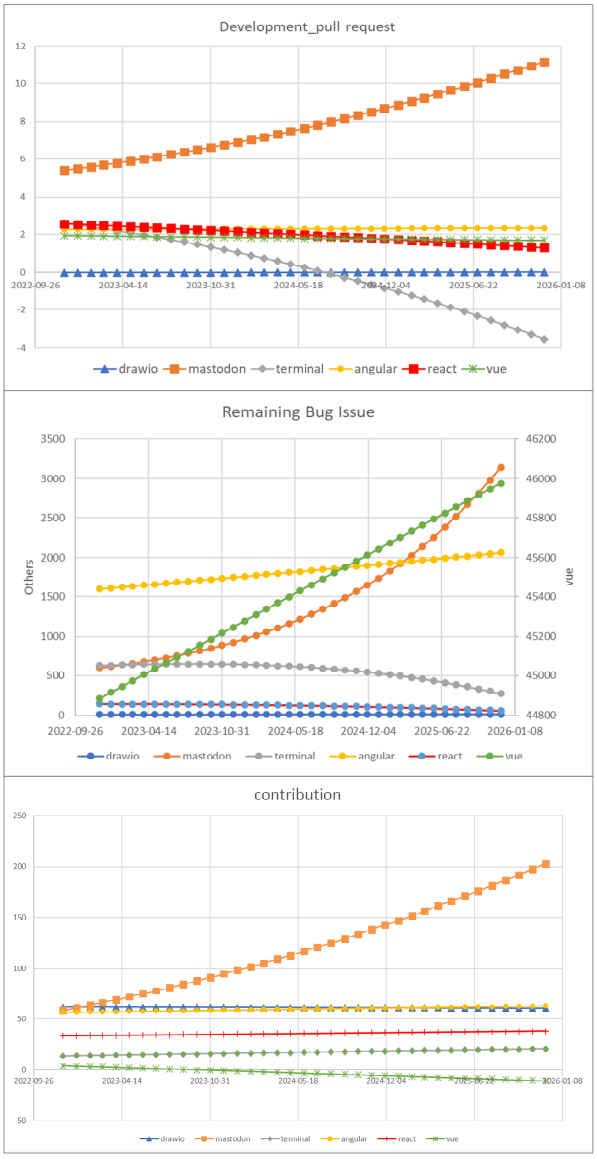

그림 2. 오픈소스 경과 예측 결과
Fig 2. Result of predicting open-sources progress

4.1에서 선택한 오픈소스들의 개발, 버그 생성 및 해결, 기여도 예측을 수행하였다. 예측 결과는 2와 같다.

식 (7)을 이용하여 오픈소스별 예측 수행 결과를 점수로 계산하였다. 개발 진행 점수(Development Score, DS), 이슈 진행 점수(Bug Issue Score, BIS), 기여 진행 점수(Contribution Score, CS)는 예측된 각 데이터들의 증가 혹은 감소 추세를 나타낸다. 오픈소스 별 DS, BIS, CS를 계산한 결과는 표 3과 같다.

$$\nabla S = \frac{\sum_j (x_j - \bar{x})(y_j - \bar{y})}{\sum_i (x_i - \bar{x})^2} \tag{7}$$

표 3. 오픈소스별 점수
Table 3. Scores of open-sources

Open-source	∇DS	∇BIS	∇CS
Mastodon	0.00525	2.16999	0.13166
Terminal	-0.00556	-0.30329	0.00604
Drawio	0.00003	0.00376	-0.00088
Angular	0.00007	0.42508	0.00562
React	-0.00114	-0.07372	0.00408
Vue	-0.00022	0.10084	-0.01293

4.3 결과 분석

그림 3과 같이 오픈소스별 DS, BIS, CS 세 값을 비교하였다. 이를 통해 각 오픈소스 별 특징을 분석하여 성장기, 정체기, 쇠퇴기 3가지로 분류하였다. 성장기는 기능 개발 및 기여자들의 참여가 활발하여 프로젝트의 규모가 커지는 시기이다. 정체기는 기능 개발 및 기여자들의 참여가 성장기보다 적으며, 프로젝트의 유지보수가 대다수인 시기이다. 쇠퇴기는 기능 개발 및 유지보수가 거의 이뤄지지 않는 시기이다. 3가지 유형의 기준은 다음과 같이 정의한다.

- 성장기: 평균 개발량> 0이고, 개발, 버그 이슈, 기여도 모두 증가
- 정체기: 평균 개발량> 0이고, 개발, 버그 이슈, 기여도 중 하나 이상 감소
- 쇠퇴기: 평균 개발량≤ 0이거나, 개발, 버그 이슈, 기여도 중 세 개 모두 감소

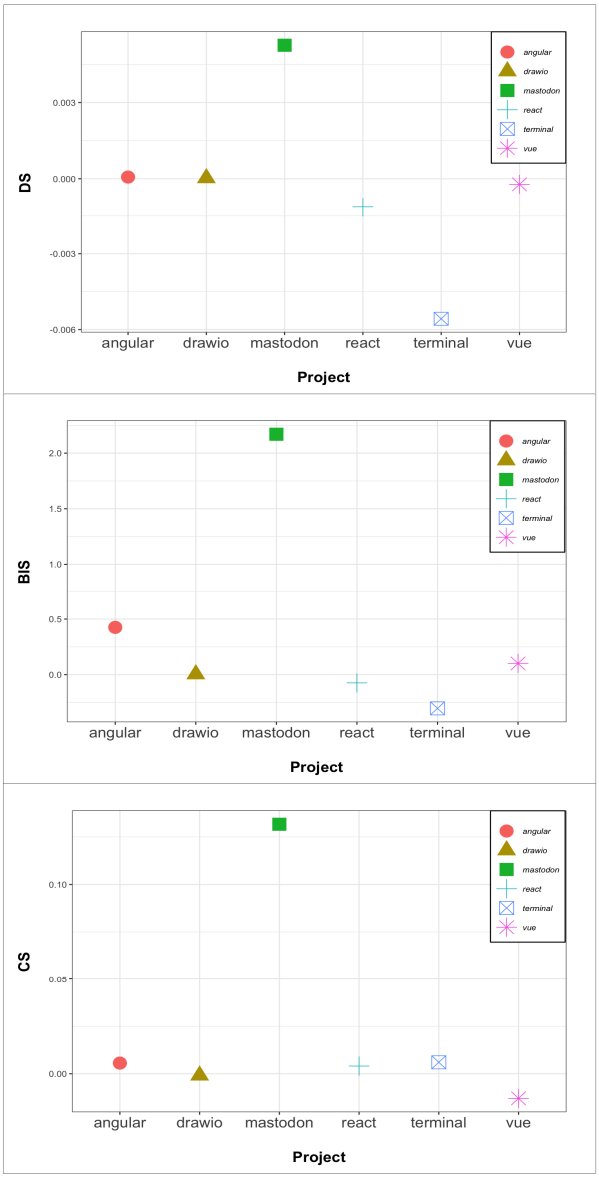

그림 3. 오픈소스별 점수
Fig 3. Scores of open-sources

위 기준의 평균 개발량 $Development_{avg}$는 식 (8)을 이용하여 계산하였다. 전체 예측 기간동안의 1번째 날의 PR_1(풀 리퀘스트)부터 n번째 날의 PR_n까지 평균을 구하였다.

$$Development_{avg} = \frac{\sum_{i=1}^{n} PR_i}{n} \quad (8)$$

위의 기준에 따라 오픈소스를 분류하였다. 분류한 결과는 표 4와 같다.

표 4. 오픈소스 분석 결과
Table 4. Result of open-sources analysis

Open-source	Type	Average of development
Mastodon	Growth	7.864
Terminal	Decline	-0.005
Drawio	Decline	-0.005
Angular	Growth	2.279
React	Stagnation	1.945
Vue	Stagnation	1.792

많은 개발량을 가졌던 Mastodon은 개발, 버그 이슈, 기여도 3가지 모두 점수가 가장 높았다. 앞으로 개발 활동과 버그 발견 및 공유 활동이 활발할뿐만 아니라 기여자들의 개발 참여율 또한 증가할 것으로 나왔다. 평균 개발량 또한 7.86으로 앞으로도 성장기에 속할 것으로 예상된다. 반면, Mastodon보다 높은 인지도를 가진 Angular, React, Terminal, Vue는 전체적으로 Mastodon보다 낮은 점수를 보였다. 높은 인지도에 많은 개발량을 가진 Angular의 경우 평균 개발량은 2.27이고, 개발 활동, 버그 발견, 기여자들의 참여 모두 증가할 것으로 예측되었다. 앞으로도 성장기에 속할 것으로 예상된다. 나머지 React, Terminal, Vue의 경우는 개발 활동이 하향세를 띌 것으로 보였다. Vue의 경우는 6개의 오픈소스 중 가장 인기도가 높은 오픈소스임에도 불구하고 CS가 가장 낮게 계산되었다. Vue와 React는 앞으로 기여자들의 참여가 낮고, 평균 개발량이 각각 1.792, 1.945여서 정체기에 들어설 것으로 예상된다. Terminal의 경우는 평균 개발량이 -0.005로 개발이 거의 되지 않을 것으로 예상되어 앞으로 쇠퇴기일 것으로 예상된다. 마지막 낮은 개발량을 가진 Drawio의 경우 CS를 제외하고 모두 0 이상이었다. 하지만 평균 개발량이 -0.005로 terminal과 같이 거의 개발이 되지 않을 것으로 예측되어 앞으로 쇠퇴기일 것으로 예상한다.

높은 개발량인 Mastodon은 성장기, 낮은 개발량인 Drawio는 쇠퇴기로 예측되었다. 그리고 높은 인기도를 가진 Angular, Terminal, React, Vue 중 3개가 정체기 혹은 쇠퇴기일 것으로 예측되었다. 인기도는 생존 가능성에 개발량보다 낮은 영향을 끼치는 모습을 보였다. 이러한 결과를 통해 높은 인기도가 높은 생존 가능성을 보장하지 않는다는 것을 알 수

있었다.

본 논문에서 예측한 결과 [4]와 다르게 기여도는 오픈소스의 생존에 핵심은 아니었다. 4.3의 예측 결과를 보면, 기여자의 수와 기여도가 제일 큰 Angular가 성장기이긴 하지만 React는 쇠퇴기로 예측되었고 기여도가 React보다 적은 Mastodon은 성장기로 예측되어 생존 가능성이 더 높을 것으로 예상한다. 기여자들의 개발 참여 즉, 기여도가 높아진다고 해도 개발량 자체는 적어지는 모습을 보였다. 또한, [5]의 연구와 달리 기여자가 오픈소스의 생존 가능성에 영향을 끼치긴 하지만 개발량이 더 많은 영향을 끼친다는 결과를 얻을 수 있었다. 또한, 코드의 복잡도를 고려하지 않아도 개발, 버그 이슈, 기여도만으로도 충분히 오픈소스의 생존 가능성을 분석할 수 있었다.

Ⅴ. 결론 및 향후 연구

현대의 소프트웨어 개발에서는 다양한 오픈소스가 사용되고 있다. 오픈소스는 개발 편의성을 향상시켰지만, 오픈소스 지원 중단으로 인한 유지보수성 저하라는 문제 또한 만들게 되었다. 이 문제를 최소화하기 위해서는 생존 가능성이 높은 오픈소스를 사용해야 한다.

본 논문에서는 지도학습의 회귀 분석을 이용하여 오픈소스의 생존 가능성을 예측하였다. 시간, 커밋, 이슈, 기여자 등 다양한 프로젝트 데이터들을 이용하여 개발 추세, 버그 생성 및 해결 추세, 기여도 추세를 예측하여 이를 통해 오픈소스의 생존 가능성을 성장기, 정체기, 쇠퇴기로 분류하였다. 제안한 방법을 토대로 오픈소스의 생존 가능성을 예측한 결과 6개의 오픈소스 중 React와 Vue는 정체기, Drawio와 Terminal은 쇠퇴기로 분류되었다. 이 생존 가능성이 낮은 4가지 프로젝트들은 개발량이 감소하거나 거의 존재하지 않은 모습을 보였다. 또한, 버그 이슈 생성 활동, 기여자들의 개발 참여 등 유저들의 해당 프로젝트 활동이 감소하는 특징을 가진 것을 확인할 수 있었다.

이러한 결과를 바탕으로 미리 프로젝트의 생존 여부를 판단하여 소프트웨어 개발 시 생존 가능성이 높은 오픈소스만 선택하여 사용할 수 있을 것이다. 이로인해, 오픈소스의 지속적 지원으로 소프트웨어의 유지보수성 향상에 기여를 할 수 있을 것이다.

본 논문에서는 오픈소스의 생존 가능성을 예측하는 방법을 제안하였다. 제안한 방법을 오픈소스 6개에 적용하여 각각의 생존 가능성을 예측하였다. 향후 연구에서는 표본을 확장시켜 생존 가능성 별 오픈소스들의 특징을 일반화하여 제안 방법의 효과를 보이고자 한다.

References

[1] H. Borges, A. Hora, and M. T. Valente, "Understanding the Factors that Impact the Popularity of GitHub Repositories", 2016 IEEE International Conference on Software Maintenance and Evolution(ICSME), Raleigh, NC, USA, pp. 334-344, Oct. 2016. https://doi.org/10.1109/ICSME.2016.31.

[2] D. Spinellis and C. Szyperski, "How IsOpen Source AffectingSoftware Development?", IEEE Software, Vol. 21, No. 1, pp. 28-33, Aug. 2004. https://doi.org/10.1109/MS.2004.1259204.

[3] N. D. Linh, P. D. Hung, V. T. Diep, and T. D. Tung, "Risk Management in Projects Based on Open-SourceSoftware", ICSCA '19: Proceedings of the 2019 8th International Conference on Software and Computer Applications, pp. 178-183, Feb. 2019. https://doi.org/10.1145/3316615.3316648.

[4] V. K. Eluri, T. A. Mazzuchi, and S. Sarkani, "Predicting long-time contributors for GitHub projects using machine learning", Information and Software Technology, Vol. 138, Oct. 2021. https://doi.org/10.1016/j.infsof.2021.106616.

[5] V. K. Eluri, S. Sarkani, and T. A. Mazzuchi, "Open Source Software Survivability Prediction Using Multi Layer Perceptron", Proceedings of 28th International Conference on Software Engineering and Data Engineering, Vol. 64, pp. 148-157, Sep. 2019. https://doi.org/10.29007/cmc6.

[6] S. Moon, S. Jang, J. Lee, and J. Lee, "Trends in

Machine Learning and Deep Learning Technologies", Information & communications magazine, Vol. 33, No. 10, pp. 49-56, Oct. 2016.

[7] H. Lee, S. Chung, and E. Choi, "A Case Study on Machine Learning Applications and Performance Improvement in Learning Algorithm", Journal of Digital Convergence, Vol. 14, No. 2, pp. 245-258, Feb. 2016. https://doi.org/10.14400/JDC.2016.14.2.245.

[8] K. Kumari and S. Yadav, "Linear Regression Analysis Study", Journal of the Practice of Cardiovascular Sciences, Vol. 4, No. 1, pp. 33-36, May 2018. https://doi.org/10.4103/jpcs.jpcs_ 8_18.

[9] E. Ostertagová, "Modelling using polynomial regression", Procedia Engineering, Vol. 48, pp. 500-506, Nov. 2012. https://doi.org/10.1016/j.proeng.2012.09.545.

[10] D. Maulud and A. Abdulazeez, "A Review on Linear Regression Comprehensive in Machine Learning", Journal of Applied Science and Technology Trends, Vol. 1, No. 4, pp. 140-147, Dec. 2020. https://doi.org/10.38094/jastt1457.

[11] H. Borges, A. Hora, and M. T. Valente, "Predicting the Popularity of GitHub Repositories", Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering, No. 9, pp. 1-10, Sep. 2016. https://doi.org/10.1145/2972958.2972966.

저자소개

박 소 희 (Sohee Park)

2018년 3월 ~ 현재 : 경기대학교 AI컴퓨터공학부 학사과정
관심분야 : 소프트웨어 공학, 소프트웨어 안전성, 머신 러닝

권 령 구 (Ryeonggu Kwon)

2011년 2월: 경기대학교 컴퓨터과학과(이학사)
2013년 2월: 경기대학교 컴퓨터과학과(이학석사)
2013년 3월 ~ 현재 : 경기대학교 AI컴퓨터과학과 박사과정
관심분야 : 모델 검증, 정형 합성, 머신 러닝, 정형 기법

권 기 현 (Gihwon Kwon)

1985년 2월 : 경기대학교 전자계산학과(이학사)
1987년 8월 : 중앙대학교 전자계산학과(이학석사)
1991년 2월 : 중앙대학교 전자계산학과(공학박사)
1991년 2월 ~ 현재 : 경기대학교 AI컴퓨터공학부 교수
2006년 ~ 2007년 : 미국 카네기멜론대학 전산학과 방문교수
2014년 ~ 2016년 : 한국정보과학회 소프트웨어공학 소사이어티 회장
2021년 ~ 현재 : 경기대학교 SW중심대학 사업단장
2022년 ~ 현재 : 경기대학교 소프트웨어경영대학 학장
관심분야 : 소프트웨어 공학, 소프트웨어 안전성