5-2023

NICHE: A curated dataset of engineered machine learning projects in Python

Ratnadira WIDYASARI

Zhou YANG

Ferdian THUNG

Sheng Qin SIM

Fiona WEE

See next page for additional authors

Citation

Author

Ratnadira WIDYASARI, Zhou YANG, Ferdian THUNG, Sheng Qin SIM, Fiona WEE, Camellia LOK, Jack PHAN, Haodi QI, Constance TAN, David LO, and David LO

NICHE: A Curated Dataset of Engineered Machine Learning Projects in Python

Ratnadira Widyasari, Zhou Yang, Ferdian Thung, Sheng Qin Sim, Fiona Wee, Camellia Lok, Jack Phan,
Haodi Qi, Constance Tan, Qijin Tay, and David Lo

School of Computing and Information System, Singapore Management University

{ratnadiraw.2020,zyang,ferdianthung,sqsim.2018,fiona.wee.2018,camellialok.2017,jack.phan.2018}@smu.edu.sg

{haodi.qi.2017,hytan.2018,qijin.tay.2018,davidlo}@smu.edu.sg

Abstract—**Machine learning (ML) has gained much attention and been incorporated into our daily lives. While there are numerous publicly available ML projects on open source platforms such as GitHub, there have been limited attempts in filtering those projects to curate ML projects of high quality. The limited availability of such high-quality dataset poses an obstacle in understanding ML projects. To help clear this obstacle, we present *NICHE*, a manually labelled dataset consisting of 572 ML projects. Based on evidences of good software engineering practices, we label 441 of these projects as engineered and 131 as non-engineered. This dataset can help researchers understand the practices that are followed in high-quality ML projects. It can also be used as a benchmark for classifiers designed to identify engineered ML projects.**

Index Terms—**Engineered Software Project, Machine Learning, Python, Open Source Projects**

I. INTRODUCTION

There are many valuable pieces of information stored in a version control system of a project; they include: source code, documentation, issue reports, test cases, list of contributors, etc. Researchers mine these software repositories to get useful insights related to how bugs are fixed [1], how developers collaborate [2] and so on. With the abundance of the open source repositories in GitHub, researchers can mine for insights and validate hypotheses on a large corpus of data. However, Kalliamvakou et al. showed that most repositories in Github are of low-quality [3], [4], which can lead to wrong and biased conclusions. To avoid skewed findings, researchers usually take some measures to filter out low-quality projects, e.g., by choosing projects with a high number of stars (which is considered to reflect the projects' popularity). Unfortunately, popularity may not be correlated with project quality [5]. Therefore, Munaiah et al. propose an approach to find high-quality software projects, more specifically; by identifying engineered software projects [6]. Such projects are essential for mining software repository (MSR) research, as they allow for high-quality findings to be uncovered (from high-quality data).

Machine learning (ML) projects are becoming increasingly popular and play essential roles in various domain, e.g., code processing [7], [8], self-driving cars, speech recognition [9], etc. Despite widespread usage and popularity, only a few research works try to examine AI and ML projects to identify unique properties, development patterns, and trends. Gonzalez et al. [10] find that the AI & ML community has unique characteristics that should be considered in future software engineering and MSR research. For example, more support is needed for Python as the main programming language, and there are significant differences between internal and external contributors in AI & ML projects. We coin a term for such research: Mining Machine Learning Repository (MLR). Similar to conventional MSR research, MLR also requires high-quality projects. In GitHub, there are many tutorials, resource pages, courseworks and toy projects that are related to ML; some of which are very popular but unsuitable for MLR research. To facilitate MLR research, we present a curated dataset of E**N**g**I**neered Ma**CH**ine L**E**arning Projects in Python (*NICHE*). We first automatically identify projects in GitHub that: (1) use one of the popular ML libraries, and (2) satisfy some basic quantitative quality metrics. This process returns 572 ML projects from GitHub. Next, we manually analyze the 572 ML projects and label them as engineered or not engineered. This dataset can be used as the raw material for MLR research, or as the benchmark for evaluating classifiers designed to identify engineered ML projects.

We label the dataset manually to ensure high quality and accurate labels. Our criteria for assessing an ML project are rooted in Munaiah et al. work [6]. We check 8 distinct dimensions of a project (architecture, community, CI, documentation, history, issues, license and unit testing) to evaluate whether the project is engineered or not. Out of the 572 projects we collected, 441 projects are labelled as engineered ML projects, and 131 projects are labelled as non-engineered ML projects. There are several related datasets in the literature. Datasets from [6] and [11] have labels indicating whether a project is engineered or not, but they do not contain ML projects. Gonzalez et al. [10] collected a dataset of ML & AI projects, but these projects are not comprehensively assessed based on their adoption of good software engineering practices. They only eliminate tutorials, homework assignments and so on. We make our dataset publicly available[1].

The rest of this paper is organized as follow. Section 2 describes the methodology used to collect and filter the dataset, as well as how the dataset is stored. Section 3 gives an overview of the dataset. In Section 4, we propose some

---

[1]https://doi.org/10.6084/m9.figshare.21967265

potential research applications that the dataset can be used for. In Section 5, we discuss some related work. Section 6 briefly mentions threats to validity. Finally, we conclude the paper and present future work in Section 7.

II. DATASET COLLECTION METHODOLOGY

In this section, we discuss how we collect, curate and store the dataset.

A. Automated Data Collection

We use GitHub as our data source. GitHub is the most active open source community and project hosting site. It provides rich information related to projects, including popularity, pull requests, statistics of commit history, dependency graph, etc. These data enable us to easily determine whether a software project adopts good software engineering practices.

This paper focuses on ML projects that use at least one of these popular ML libraries: Theano,[5] PyTorch,[6] and TensorFlow.[7] GitHub provides dependency graphs which lists all the public projects that depend on a library. We fetch lists of projects from the 3 ML libraries' dependency graphs and also get some statistical information, e.g. number of stars, number of commits and timestamp of the latest commit. To filter out low-quality projects, we only consider projects that satisfy the following requirements:

1) **Popularity**: the project must have 100 or more stars, as we want to filter toy projects that are clearly non-engineered.
2) **Activity**: the project must have at least one commit that is more recent than May 1 2020, as we want to analyze projects that are still updated when we collecting the data.
3) **Project Availability**: the project should be public and accessible via GitHub API.
4) **Commit History**: the project have at least 100 commits. We use this as a threshold to filter out new projects or projects that only have a short history.

After applying these filters, we have 572 projects for further analysis.

B. Manual Data Curation

To ensure the quality of the dataset, we filter and label the ML projects manually. We adopt the framework used by Munaiah et al. [6] to decide whether an ML project is an engineered project or not. In this framework, a software project is considered to be engineered if it there is a clear evidence of sound software engineering practices such as good architecture, good documentation, sufficient testing, evidence of proper project and so on. Here we list the eight dimensions Munaiah et al. [6] used to assess a software project:

- **Architecture**: Well-defined relationships between components within a project is an indicator of good code organization.

- **Community**: Presence of many collaborators in a project shows that there is a form of collaboration during the project development.
- **Continuous Integration (CI)**: Using CI ensures that the code is stable for development or release, which indicates that the project has a good code quality.
- **Documentation**: The presence of sufficient documentation shows developers' thinking of code maintainability.
- **History**: When the code have undergone continual changes or sustained evolution, it indicates that the project is being modified to ensure its viability.
- **Issues**: Projects that maintain management activities such as requirements, schedules, and tasks through GitHub Issues show proper project management.
- **License**: Having license in the project, explicitly shows the rights of the user in making copies of the project, which shows a good code accountability.
- **Unit testing**: Presence of testing in projects shows the developer's effort to ensure that the project work as it should, indicating that the project is of good quality.

There are no fixed criteria for deciding to what extent fulfilling these dimensions makes a project be engineered. But, in general, fulfilling one or two of these dimensions does not guarantee that the project is engineered. However, having the majority of the dimensions fulfilled makes the project likely to be engineered.

As the projects that we assess are still actively updated, it is possible that the status and label for these projects change over time. For our analysis, we collected the projects on June 2020. Each project is assessed by two authors who independently evaluate each dimension and give their final judgements. The authors check if the project satisfies the dimensions by looking at its code, documentation, contributors, issues, and other information available in GitHub. The following are guidelines that authors use to assess a project: (1) look for evidence of sufficient testing, e.g. number of files related to test and code coverage (if applicable); (2) look for evidence that the project has architectural decisions in organizing the code, e.g. the code is split into different modules and no ad-hoc scripts; (3) check whether good documentations are provided; (4) check if the project uses issues to track new features and bugs; (5) check if the project uses a CI service, e.g. Travis, CircleCI, etc; (6) check if the project is updated within the last one month; (7) check how many active contributors the project has; (8) check whether the project provides a license.

For every point in the guideline, we consider the following dimensions for the project assessment: unit testing for point (1), architecture for point (2), documentation for point (3), issues for point (4), CI for point (5), history for point (6), community for point (7), and license for point (8). Aside from providing a label on whether a project is engineered or not, the labellers also provide descriptive information for every dimension. In case both labellers do not agree (on whether a project is engineered or not), the disagreement is resolved by a discussion. This discussion is led by the author that does not label the project and is more experienced.

	OpenNMT/OpenNMT-tf	mesutpiskin/computer-vision-guide
Architecture	The source code of this project has been organized into different modules in a way that shows the relationship between different files.	The source code of this project has been organised into different folders.
Community	20 actives contributors found in https://github.com/OpenNMT/OpenNMT-tf/graphs/contributors.	There are no contributors to the project other than the author.
Continuous Integration	The project uses Travis CI[2] to hosted continuous integration service.	Continuous Integration (CI) is not used on the project.
Documentation	There is a well-maintained README, and the majority of the functions i.e.: runner.py, training.py, and text.py have appropriate docstrings.	Most of the functions in the source code are not documented. Only have the general information regarding project on the README.md.
History	20 commits in June 2020 as seen in their pulse monthly page out of total 1,735 commits from July 2017.	2 commits in June 2020 as seen in their pulse monthly page out of total 101 commits from September 2018
Issues	The project maintainers respond to issues, as evidenced by issues that were quickly resolved by the developers[34].	There is no issues raised or closed from May 2019. Only has one issues and the authors take more than 6 months to close this issue.
License	Project license is clearly written in their GitHub page.	Project license is clearly written in their GitHub page.
Unit Testing	There are many files related to testing.	There is no evidence of any testing done on the project.

An example of a project that is labelled as engineered is shown in Table I on the left column. In this example, the `OpenNMT/OpenNMT-tf`[8] project is labelled as an engineered ML project since it satisfies all the dimensions in the evaluation framework. For example, there is evidence of quick response to issues, usage of CI service, sufficient testing and so on. Unlike the engineered ML project, the non-engineered ML project shown on the right column of Table I does not fulfil enough dimensions. The mesutpiskin/computer-vision-guide[9] is labelled as a non-engineered ML project because most of the dimensions in this project are not satisfied. For example, this project does not have an active community contributor, as the only contributor is the author himself. The project also does not have any test cases or employ CI, does not have much documentation, and only has one issue since the creation of the project. The only dimension that is satisfied by the project is license.

Labelling this dataset takes around 123 person-hours in total. Out of the initial 572 projects that we collected using GitHub API, 441 projects are labelled as engineered ML projects and 131 projects are labelled as non-engineered.

C. Data Storage

Our dataset is available at https://github.com/soarsmu/NICHE. We provide a CSV file that contains the list of the project names along with their labels and descriptive information for every dimension. We also provide several basic statistics for each project, such as the number of stars and commits.

III. DATASET OVERVIEW

In this section, we provide some basic statistics, including the number of stars, commits and lines of code of the 572 projects in our dataset.
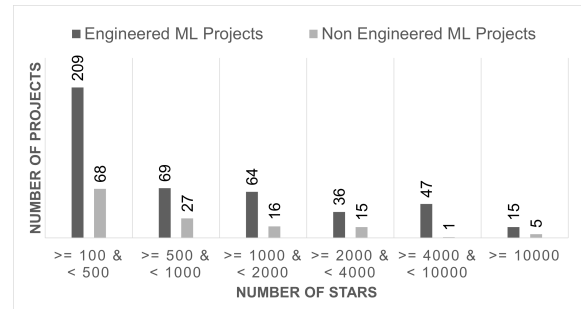
[8]https://github.com/OpenNMT/OpenNMT-tf

[9]https://github.com/mesutpiskin/computer-vision-guide

Fig. 1. Distribution of projects in terms of number of stars

A. Number of Stars

The number of stars in GitHub reflects how popular the corresponding project is. The number of stars has been shown to positively correlate with the popularity of the project [12], [13]. We count the number of stars for each project in our dataset. In Figure 1, we group projects based on their numbers of stars. We observe that around half of the projects (280 out of 572) have between 100 and 500 stars. Around 75.45% projects in this category are labelled as engineered ML projects. We also observe that projects with more stars are more likely to be engineered, but the inverse does not hold as only 47.5% engineered projects have 100 to 500 stars. This observation is similar to previous findings by Munaiah et al. [6].

B. Number of Commits

The number of commits reflects a project development history. In Figure 2, we group projects based on their numbers of commits. One interesting pattern is that the more commits a project has, the more likely the project is an engineered ML project. This pattern is evidenced by the fact that the ratio of engineered ML projects increases as the number of commits increases.
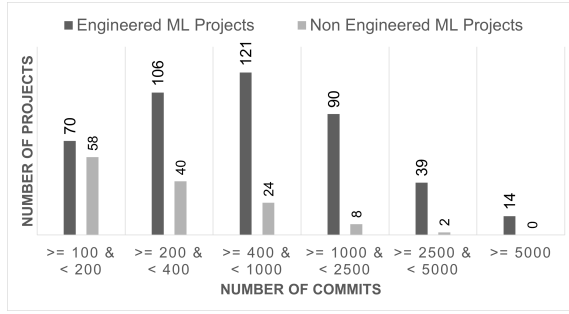
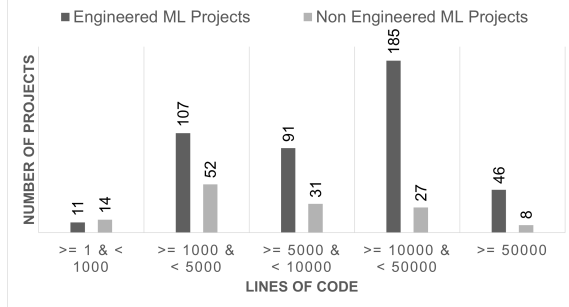Fig. 2. Distribution of projects in terms of number of commits

Fig. 3. Distribution of projects in terms of lines of code

C. Lines of Code (LOC)

The number of Lines of Code (LOC) can be an indicator of project size. We clone all the collected projects and checkout their main branch. Then, we use cloc[10] to count LOC. We plot the distribution of projects' LOC in Figure 3. The largest project has 705,797 LOC and is labelled as engineered. For projects with less than 1,000 LOC, the number of non-engineered projects surpasses that of engineered ML projects. We observe that the ratio of engineered ML projects increases as the number of LOC increases.

IV. POTENTIAL RESEARCH APPLICATION

Our dataset can be used to explore many research topics such as mining machine learning repositories and evaluating engineered ML project classifiers.

Mining Machine Learning Repositories. As machine learning projects become more popular and important, many researchers start to analyze properties of machine learning projects [10], [14]–[16], which we refer to as Mining Machine Learning Repository (MLR). High-quality ML projects are essential for MLR. Some research works were done on a few projects, e.g. Peng et al. find that Apollo's overall code coverage is not high [17]. Our dataset can enable researchers to validate a hypothesis at a larger scale. The dataset can be used to answer questions like how well engineered ML projects are covered by existing test cases, and what are the differences between the coverage of ML files and non-ML files in engineered ML projects.

Benchmark for engineered ML project classifiers. Reaper [6] and PHANTOM [11] both show decent performance on classifying whether a software project is engineered or not. But the datasets that they are evaluated on contain only conventional software projects. There is no intersection between their dataset and ours. How their engineered software project classifiers perform on ML projects are unknown. The community may need classifiers specially designed for ML projects, and our dataset can be the benchmark for such classifiers.

V. RELATED WORK

We identify three datasets that are most related to our work. One is the dataset from Munaiah et al. [6]. Munaiah et al. provide a dataset of 800 manually labelled software projects. 400 of them are labelled as engineered and the other 400 are labelled as non-engineered. Another related dataset is from [11] which extends Munaiah et al.'s dataset [6] with additional 200 projects. There is no overlap between their datasets and ours. Gonzalez et al. [10] provide a list of ML-related projects. However, they only eliminate projects that are tutorials, homework assignments, coding challenges, 'resource' storages, or collections of model files/code samples. These projects have not been assessed based on the criteria of good software engineering practices that are considered by Munaiah et al.

VI. THREATS TO VALIDITY

To ensure the quality of our dataset, we manually curate ML projects. However, even though we have tried our best, it is still possible that we mislabel some projects. To mitigate potential mislabelling, the data is labelled independently by two authors. If there is any disagreement, a more experienced author leads a discussion to resolve the disagreement.

VII. CONCLUSION AND FUTURE WORK

To conclude, we present *NICHE*, a dataset of ML projects in Python consisting of 441 projects that are identified as engineered ML projects and 131 projects that are identified as non-engineered ML projects. Our dataset contains the largest number of engineered ML projects in Python. The dataset is curated by hand to better ensure the correctness of the label assigned to each project. We believe the 441 engineered projects will make it easier for MSR researcher to get sizeable and high quality ML projects for their research, as it is very important to exclude poor quality projects. Moreover, the 131 non-engineered projects serve as negative examples that can be used to build a benchmark to build and evaluate a classifier that can distinguish engineered from non-engineered ML projects.

As future work, we also plan to investigate the properties of the engineered ML repositories to identify their unique characteristics, pain points faced by their contributors, and ways for researchers to help them.

REFERENCES

[1] M. Soto, F. Thung, C.-P. Wong, C. Le Goues, and D. Lo, "A deeper look into bug fixes: Patterns, replacements, deletions, and additions," in *Proceedings of the 13th International Conference on Mining Software Repositories*, ser. MSR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 512–515. [Online]. Available: https://doi.org/10.1145/2901739.2903495

[2] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, "Network structure of social coding in github," in *2013 17th European Conference on Software Maintenance and Reengineering*, March 2013, pp. 323–326.

[3] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 92–101. [Online]. Available: https://doi.org/10.1145/2597073.2597074

[4] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. Germán, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, pp. 2035–2071, 2015.

[5] H. Sajnani, V. Saini, J. Ossher, and C. V. Lopes, "Is popularity a measure of quality? an analysis of maven components," in *2014 IEEE International Conference on Software Maintenance and Evolution*, Sep. 2014, pp. 231–240.

[6] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Softw. Engg.*, vol. 22, no. 6, p. 3219–3253, Dec. 2017. [Online]. Available: https://doi.org/10.1007/s10664-017-9512-6

[7] J. He, B. Xu, Z. Yang, D. Han, C. Yang, and D. Lo, "Ptm4tag: Sharpening tag recommendation of stack overflow posts with pre-trained models," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, ser. ICPC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–11. [Online]. Available: https://doi.org/10.1145/3524610.3527897

[8] C. Yang, B. Xu, J. Khan, G. Uddin, D. Han, Z. Yang, and D. Lo, "Aspect-based api review classification: How far can pre-trained transformer model go?" in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Los Alamitos, CA, USA: IEEE Computer Society, mar 2022, pp. 385–395. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SANER53432.2022.00054

[9] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[10] D. Gonzalez, T. Zimmermann, and N. Nagappan, "The state of the ml-universe: 10 years of artificial intelligence and machine learning software development on github," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 431–442. [Online]. Available: https://doi.org/10.1145/3379597.3387473

[11] P. Pickerill, H. J. Jungen, M. Ochodek, M. Mackowiak, and M. Staron, "PHANTOM: curating github for engineered software projects using time-series clustering," *Empir. Softw. Eng.*, vol. 25, no. 4, pp. 2897–2929, 2020. [Online]. Available: https://doi.org/10.1007/s10664-020-09825-8

[12] H. Borges and M. Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 09 2018.

[13] V. Cosentino, J. L. Cánovas Izquierdo, and J. Cabot, "A systematic mapping study of software development with github," *IEEE Access*, vol. 5, pp. 7173–7192, 2017.

[14] J. Han, S. Deng, D. Lo, C. Zhi, J. Yin, and X. Xia, "An empirical study of the dependency networks of deep learning libraries," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 868–878.

[15] H. Jebnoun, H. Ben Braiek, M. M. Rahman, and F. Khomh, "The scent of deep learning code: An empirical study," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 420–430. [Online]. Available: https://doi.org/10.1145/3379597.3387479

[16] Y. Fan, X. Xia, D. Lo, A. E. Hassan, and S. Li, "What makes a popular academic ai repository?" *Empirical Software Engineering*, vol. 26, no. 1, pp. 1–35, 2021.

[17] Z. Peng, J. Yang, T.-H. P. Chen, and L. Ma, *A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems: A Case Study on Apollo*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1240–1250. [Online]. Available: https://doi.org/10.1145/3368089.3417063