UNIVERSITY OF SALERNO

Department of Computer Science

Master in Computer Science

(NICHE-23) Revised Dataset of Engineered Machine Learning for enhanced curation

SUPERVISOR

Prof. **Dr. Damian A. Tamburri**

Associate Professor,

TU/e

STUDENT

Bilal Haider

0522501075

Abstract

The NICHE-23 dataset emerges as an advanced and enriched iteration of the original NICHE collection, specifically tailored to meet the escalating demand for high-quality datasets in machine learning (ML) project curation. Encompassing 630 meticulously selected projects—introducing 60 new ones while excluding 2 that became obsolete—NICHE-23 stands as a more expansive and current resource for both researchers and practitioners. This version updates each project with the latest metrics on Stars, commits, Lines of Code, and issue resolutions, ensuring a dataset curated against rigorous criteria across eight distinct dimensions. With 483 projects distinguished as "Engineered" ML endeavors and 147 as "non-engineered," NICHE-23 not only broadens the scope of curated ML datasets but also offers deep insights into prevailing practices, thereby setting a new benchmark for the development of classifiers and advancing research methodologies in the field.

Contents

List of Figures

CHAPTER 1

Introduction

In the landscape of software development, version control systems serve as rich repositories of valuable information, encompassing source code, documentation, issue reports, test cases, and contributor lists. Researchers extensively mine these repositories to glean insights into bug-fixing methodologies, developer collaboration dynamics, and other pertinent aspects of software engineering [1]. Platforms like GitHub offer a vast array of open-source repositories, providing researchers with data to validate hypotheses and draw meaningful conclusions. However, studies by Kalliamvakou et al. have highlighted that a significant portion of GitHub repositories are of low quality, potentially skewing research findings and leading to biased conclusions. To mitigate this issue, researchers often employ various measures to filter out low-quality projects, such as selecting projects with a high number of stars [2]. However, the correlation between popularity and project quality remains tenuous, as noted by Munaiah et al., who propose an approach to identify high-quality software projects by focusing on engineered projects.

The rapidly growing popularity of machine learning (ML) projects across diverse domains such as code processing, self-driving cars, and speech recognition, there exists a gap in research examining the unique properties, development patterns, and trends within the AI and ML community. While some studies have shed light on

these aspects, including the work by Gonzalez et al., which underscores the need for tailored support for Python and highlights disparities between internal and external contributors in AI and ML projects, there remains a dearth of curated datasets specifically tailored for Mining Machine Learning Repository (MLR) research [3].
To address this gap, we introduce NICHE-23, an enhanced and meticulously curated dataset of Engineered Machine Learning Projects in Python. Building upon previous efforts, NICHE-23 comprises a comprehensive collection of ML projects sourced from GitHub, selected based on their utilization of popular ML libraries and adherence to basic quantitative quality metrics. Through manual analysis and labeling, we categorize these projects as engineered or non-engineered, employing criteria rooted in established software engineering practices.

This introduction sets the stage for the subsequent sections, where we delve into the methodology used to collect and filter the dataset, provide an overview of NICHE-23, review related work, research methodology, address research questions, discussion on results and conclude with insights into future endeavors. Through NICHE-23, we aim to facilitate MLR research by providing a curated repository of high-quality ML projects, thereby fostering robust and meaningful investigations within the machine learning community.

CHAPTER 2

Related Work

The landscape of curated datasets in the domain of machine learning (ML) and software engineering provides valuable context for understanding the evolution from the previous NICHE dataset to NICHE-23, the updated version. Specifically, three datasets are most related to our work: the dataset from Munaiah et al. [4] and it's extension presented in [5], the ML-related projects list provided by Gonzalez et al. [6] and NICHE: a curated dataset by R. Widyasari et al. [7]

Munaiah et al. [4] introduced a dataset comprising 800 manually labeled software projects, with 400 labeled as engineered and the remaining 400 as non-engineered. This dataset served as a foundational reference for assessing project quality based on established software engineering practices. Subsequently, an extension of this dataset was presented in [5], which added an additional 200 projects. Notably, there is no overlap between the NICHE dataset.

Gonzalez et al. [6] compiled a list of ML-related projects; however, their focus was primarily on eliminating projects such as tutorials, homework assignments, coding challenges, and collections of model files/code samples. This dataset has not been assessed based on the criteria of good software engineering practices considered by Munaiah et al.[4] and subsequently adopted in NICHE-23.

R. Widyasari et al. [7] comprised the curated dataset total of 572 Engineered/non-engineered ML projects on Github. In which 441 projects are labelled as engineered ML projects, and 131 projects are labelled as non-engineered ML projects which framework used by Munaiah et al. [4] to decide whether an ML project is an engineered project or not. The transition from the previous NICHE dataset to NICHE-23 involved the addition of 60 new entries and the removal of 2 outdated projects, resulting in a revised dataset comprising a total of 630 projects. This update represents an effort to enhance the comprehensiveness and relevance of the dataset, ensuring that NICHE-23 remains a valuable resource for researchers and practitioners in the field of ML and software engineering.

CHAPTER 3

Research Methodology

This section involves the method proposed by Widyasari et al. [7] to collect, curate, store the data and also modify the dataset already stored in NICHE. The dataset was originally put together in June 2020, but I revisited it in December 2023 to make sure it stayed relevant.

The methodology employed for modifying the dataset involved a meticulous manual review of each repository included in the dataset. This process required examining each project individually to verify its current status and update relevant fields. By conducting a thorough review of each repository, the research methodology ensured that the dataset remained current and reflective of the evolving landscape of machine learning projects on GitHub.

3.1 Dataset Modification

Before adding any new machine learning projects created in the past three years, I made some changes to the existing dataset. This included updating information like how popular each project is (measured by Stars), how many times its code has been updated (Number of Commits), and how much code it contains (Lines of Code). I also checked to see if each project was still available on GitHub and if it still met the criteria for being considered a good quality machine learning project.

3.1.1 Updating Existing Data

Stars and Commits: We carefully checked each project in the dataset, one by one, to see how many Stars it had and how many times its code had been updated (Commits). By doing this manually, checking each repository. The stars and commits are updated. We made sure to get the most recent numbers for each project. This helped keep the dataset accurate and up to date, making it more useful for research in machine learning.

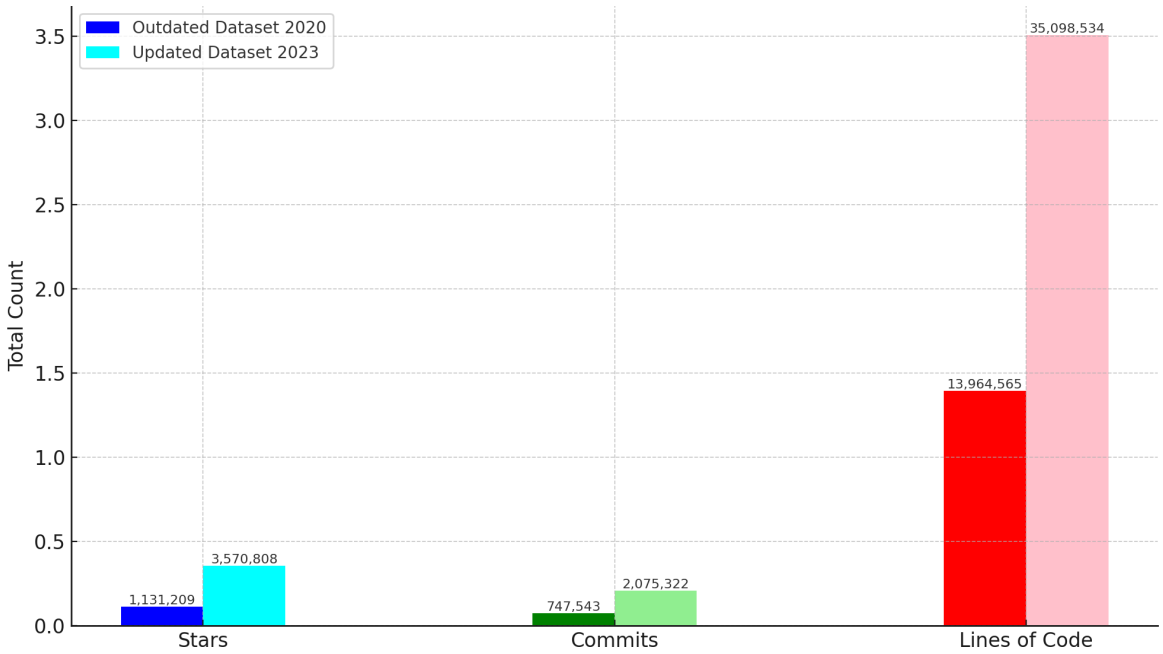

Figure 3.1: Comparison Graph Of Datasets

The Figure 3.1 illustrates a significant leap in the total stars from approximately

1.13 million to 3.57 million, demonstrating an increase in the perceived value of the projects within the dataset. In terms of commits, the updated dataset showcases an uptick from around 747,000 to over 2 million, underscoring a heightened level of development activity. The precision in updating these metrics ensures that the NICHE dataset remains a relevant and accurate resource, reflecting the latest trends and activities within the machine learning landscape on GitHub. This diligence is instrumental for researchers relying on the dataset for studies in software engineering practices and the evolution of machine learning projects.

Lines of Code (LOC): Revising the lines of code plays a crucial role in updating the dataset. Which is done by using the technique advised by R. Widyasariet al. [7] NICHE: the curated dataset. This process involves utilizing the tool CLOC to clone the collected projects by using the command (git clone –depth 1 <GIT URL>) on Git Bash and switch to their main branch. Then, after the cloning, then Lines of code can be achieved by using the command (cloc <repo name>) as shown in Figure 3.2 below. Projects that contain fewer than 1,000 lines of code are categorized as non-engineered.

```
MINGW64:/c/Users/Bilal

Bilal@DESKTOP-6QMHOKT MINGW64 ~
$ git clone --depth 1 https://github.com/Unity-Technologies/marathon-envs.git
Cloning into 'marathon-envs'...
remote: Enumerating objects: 866, done.
remote: Counting objects: 100% (866/866), done.
remote: Compressing objects: 100% (549/549), done.
remote: Total 866 (delta 413), reused 618 (delta 308), pack-reused 0
Receiving objects: 100% (866/866), 161.66 MiB | 3.59 MiB/s, done.
Resolving deltas: 100% (413/413), done.
Updating files: 100% (795/795), done.

Bilal@DESKTOP-6QMHOKT MINGW64 ~
$ cloc marathon-envs
     711 text files.
     242 unique files.
     553 files ignored.

github.com/AlDanial/cloc v 1.98  T=8.93 s (27.1 files/s, 6374.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Unity-Prefab                    31              0              0          26135
C#                             154           2911           3295          19457
XML                             29            174            133           2266
Markdown                        11            169              0            750
Python                           6            155            168            676
YAML                             1             16             15            228
JSON                             6              0              0            133
Dockerfile                       1             16             12            110
INI                              1             12             11             51
C                                1              2              2              6
Text                             1              0              0              2
-------------------------------------------------------------------------------
SUM:                           242           3455           3636          49814
-------------------------------------------------------------------------------
```

Figure 3.2: LOC through CLOC Tool

3.1.2 Manual Review and Quality Assurance

A comprehensive manual review was conducted for each project to ascertain its current status on GitHub and its compliance with quality criteria. This meticulous process was essential for ensuring that all projects included in the dataset adhered to high standards of software engineering and were reflective of the latest developments in machine learning project practices.

Some of the guidelines used by Munaiah et al. [4] were (a) repository contains sufficient documentation to enable the project contained within to be used in a general-purpose setting, (b) repository contains an application or service that is used

by or has the potential to be used by people other than the developers, (c) repository does not contain cues indicating that the source code contained within may be an assignment.

3.1.3 Data Flow Diagram

The Data Flow Diagram (DFD) provides a visual representation of the workflow and processes involved in enhancing the dataset. The DFD outlines the systematic approach taken to ensure the updated dataset reflects the most current and accurate information. As shown in Figure 3.3

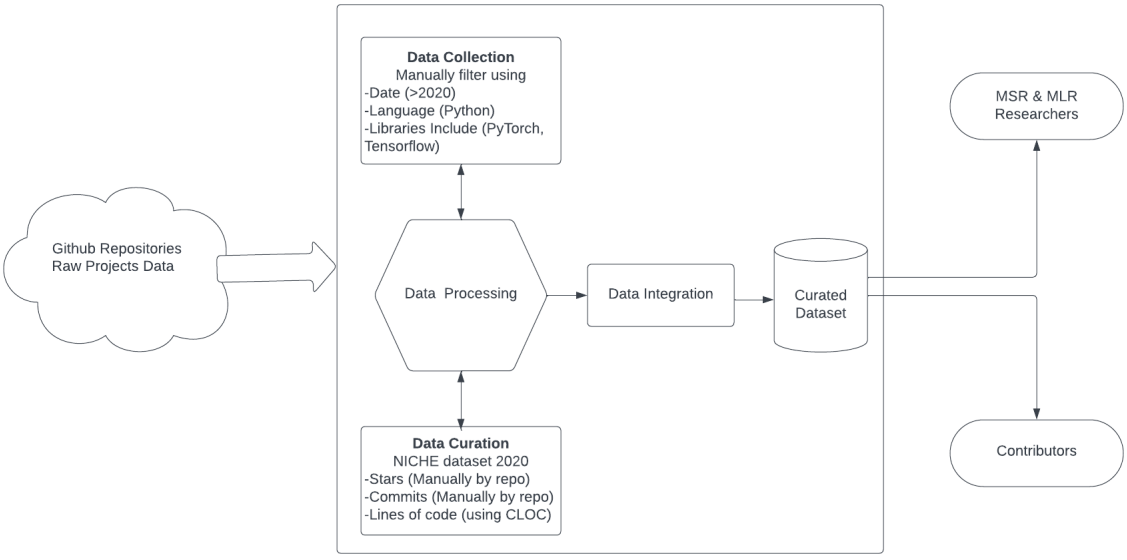

Figure 3.3: Dataflow of updated dataset

1. **Data Collection:** The first stage involves collecting raw project data from GitHub repositories. The selection criteria were based on the project's language (Python) and the usage of specific libraries (PyTorch, TensorFlow) which are indicative of active and relevant ML projects.

2. **Data Processing:** The collected data undergoes a rigorous process of verification and updating. This includes manually checking each project for the number of stars, commits, and lines of code (LOC), which are key indicators of a project's quality and engagement level.

3. **Data Curation:** Using the CLOC tool, the lines of code for each project are calculated, which plays a pivotal role in determining the project's classification as engineered or non-engineered. Projects with fewer than 1000 lines of code are generally categorized as non-engineered.

4. **Data Integration:** Once the projects have been curated, the data is integrated into a consolidated dataset. This curated dataset is then made available to MSR and MLR researchers, who can utilize it for their research, and contributors, who can add value through their expertise.

5. **MSR and MLR Researchers and Contributors:** The final entities in the DFD are the researchers and contributors who will utilize the curated dataset. Researchers can derive insights into software engineering practices in ML projects, while contributors can use the dataset to propose improvements or validate their own work.

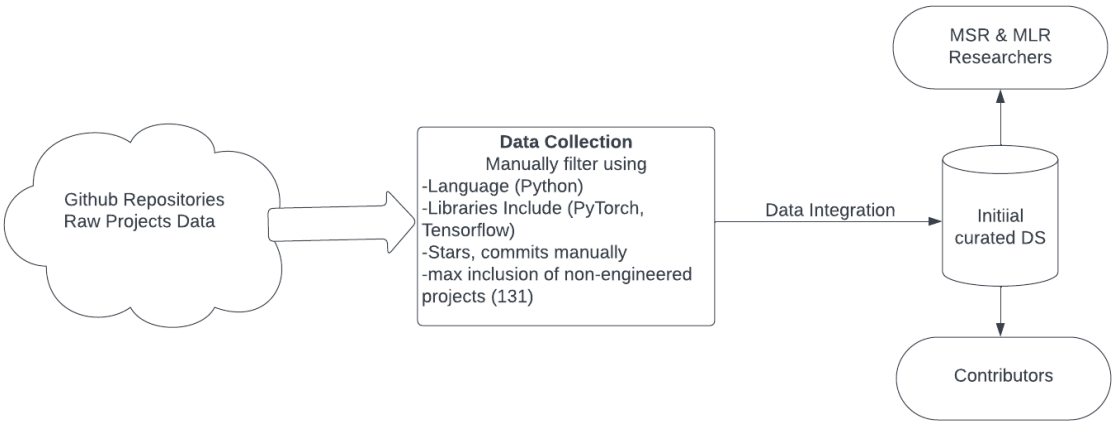

Figure 3.4: Dataflow of outdated dataset

Whereas, the above mentioned Figure 3.4 data flow diagram visualizes and reflects the functionality of old dataset which clearly shows there is only the process of data collection which was followed to collect data. Even the architecture of both datasets are same to categorized the projects engineered or non-engineered but the researchers added more non-engineered project total of 131 which some how decreases the accuracy of dataset as compared to new one as discussed in result and discussion section with test results mentioned after training the dataset into classifier.

Overall, 1372 changes are made to the dataset, including updating data and removing two projects that were no longer on GitHub. Despite all these changes, the projects still met the 8- dimensions framework or most of dimensions standards set for being considered high-quality machine learning projects [7].

3.2 Dataset Expansion

In the latest update to the NICHE dataset, a significant augmentation has been undertaken with the addition of 60 new entries, bringing the total to 630 projects. This expansion is not merely quantitative; it substantially enriches the dataset's diversity and depth, ensuring a broader representation of scenarios, technologies, and methodologies in machine learning applications.

To maintain the dataset's integrity, each machine learning (ML) project undergoes a manual filtering and labeling process. Special framework is adopted which is proposed by Munaiah et al. [4] and also applied by R. Widyasari et al. [7] to determine the engineering status of an ML project. According to this methodology, a software project is classified as engineered if it demonstrates clear application of robust software engineering principles, including good architecture, good documentation, sufficient testing, evidence of proper project and so on. Here is the list of eight dimensions used to assess a software project:

Architecture: The presence of well-defined relationships among project components signifies excellent code organization. Software architecture comes in all shapes, sizes, and complexities. There is no one fixed architecture to which all software systems adhere; however, software systems that employ some form of architecture have discernible relationships between components that compose the system [5].

Community: A high number of collaborators indicates active contribution throughout the project's development. The presence of a developer community shows that there is some form of collaboration and cooperation involved in the development of the software system, which is partial evidence for the repository containing an engineered software project.

Continous Integration (CI): The adoption of CI highlights the project's commitment to maintaining stable and high-quality code for development or release. Continuous integration (CI) is a software engineering practice in which developers regularly build, run, and test their code combined with code from other developers. The practice of continuously integrating changes ensures that the software system contained within these constantly evolving source code repositories is stable for development and/or release.

Documentation: Software developers produce and manage different types of documentation. Some documentation types are embedded within the source code, like comments in the code, while others exist outside of the source files, such as wikis,

requirements specifications, and design descriptions. Comprehensive documentation reflects the developers' foresight into ensuring code maintainability.

History: Eick et al.[8] demonstrated that source code requires ongoing modification to prevent feature deficiency and stay competitive in the market. Such modifications can include fixing bugs, adding new features, conducting preventive maintenance, resolving vulnerabilities, and more. The continuous adaptation of the software system serves as an indication of efforts to maintain its relevance and functionality. This ongoing process of change provides partial proof that the software system undergoes engineering.

Issues: Utilizing GitHub Issues for organizing requirements, timelines, and tasks demonstrates diligent project management. we assume the sustained use of the GitHub Issues feature to be indicative of management in a source code repository.

License: A project's licensing explicitly defines user permissions and responsibilities, indicating a commitment to transparency and accountability in code usage. The presence of a software license is necessary but not sufficient to indicate a repository contains an engineered software project according to our definition of the dimension.

Unit Testing: Implementing tests demonstrates the developers' commitment to ensuring the project performs correctly, signifying quality assurance. The existence of testing within a software project suggests that developers have dedicated time and effort towards verifying that the product behaves as expected. Nonetheless, simply having tests in place does not automatically indicate that the software project has been engineered. It is also important to evaluate the comprehensiveness and effectiveness of these tests [4].

Generally, meeting just a couple of the specified criteria does not automatically qualify a project as engineered. Yet, when a project aligns with most of these criteria, it strongly suggests that it has been engineered. The evaluation of whether a project meets these dimensions is conducted through a thorough review of its code, doc-

umentation, contributor activity, issue tracking, and other relevant data hosted on GitHub.

3.3 Dataset Storage

This revised dataset is available at https://github.com/bilalhaidr/NICHE-23. 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.

3.4 Research Questions

A brief introduction that reiterates the importance of the research questions and sets the stage for the discussions that follow. Mention how these questions guide the research objectives and methodology. **What are the criteria used to categorize a machine learning project as “engineered” or “non-engineered” within the NICHE dataset?**

A project within the NICHE dataset is classified as "engineered" if it demonstrates adherence to the majority of these criteria, includes architecture, community involvement, continuous integration, documentation, project history, issue management, licensing, and unit testing which showcasing a comprehensive application of software engineering principles. Conversely, projects that meet only a few or none of these criteria are considered "non-engineered," indicating potential areas for development in terms of engineering practices. This classification system helps in differentiating projects based on their engineering quality, guiding researchers and practitioners in their selection of robust, well-maintained, and effectively managed ML projects.

How does the modification of the updated dataset impact the overall structure of the NICHE dataset?

This update not only reflects the latest advancements in the field, making the dataset more relevant and up-to-date, but also improves its robustness and the granularity of analysis it supports. With a larger, more varied pool of projects, the dataset offers richer insights into engineering practices in ML projects, providing a more

solid foundation for research, educational purposes, and practical applications. This expansion thereby increases the NICHE dataset's utility as a comprehensive resource for exploring the evolving landscape of machine learning and software engineering.

How this dataset will be useful for Mining Software Repository (MSR) research and Mining Machine Learning Repository (MLR) research?

The NICHE dataset, is a goldmine for researchers looking to understand how to build better ML software and what makes these projects successful. It's like a detailed map for both software experts and machine learning enthusiasts, showing them what works well and what doesn't in real-world projects [9]. This dataset helps in comparing different projects to see common challenges and solutions, studying the use and effectiveness of various ML techniques, and exploring how projects handle their data from start to finish. It's not just for the experts; it's also a great learning tool for students and newcomers, helping bridge the gap between theory and practice. By offering a snapshot of the current state of ML projects, the NICHE dataset is invaluable for pushing forward new discoveries and improving how software and ML models are developed and maintained [10].

CHAPTER 4

Results And Discussion

The original dataset had 572 projects from which 441 of these projects are labelled as engineered and remaining 131 as non-engineered. The enhancement of the NICHE dataset to version NICHE-23, with a total of 630 projects including 60 new entries and the removal of 2 outdated projects which are removed from Github. Revising the latest data for existing projects, including Stars, commits, Lines of Code (LOC) and actively issues resolution marks a significant expansion and update. The dataset now offers a more comprehensive view of machine learning (ML) projects, with each project's data updated to reflect the active issue resolution. This curation process ensured the inclusion of projects that meet defined criteria across all 8 dimensions or most of dimensions, resulting in 483 projects classified as "Engineered" and 147 as "non-engineered." This classification is based on the presence of sound software engineering practices.

TABLE I
ENGINEERED ML PROJECT IN PYTHON: OPENNMT/OPENNMT-TF (LEFT) & NON ENGINEERED ML PROJECT IN PYTHON: MESUTPISKIN/COMPUTER-VISION-GUIDE (RIGHT)

	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[3][4].	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.

Figure 4.1: Example of a project categorization

Table I image illustrates an example of a project categorized as engineered on the left column. In this case, the OpenNMT/OpenNMT-tf project is recognized as an engineered machine learning (ML) project because it meets all the criteria outlined in the evaluation framework. This includes evidence of prompt issue responses, utilization of continuous integration (CI) services, comprehensive testing, etc. Conversely, the project listed in the right column of Table I does not meet sufficient criteria to be considered engineered. The project by mesutpiskin/computer-vision-guide is identified as a non-engineered ML project due to its failure to satisfy most of the established dimensions. For instance, it lacks a vibrant community of contributors, with the sole contributor being the author. Moreover, the project is missing test cases, does not use CI, has limited documentation, and has only one issue reported since its inception. The sole criterion it fulfills is having a license.

4.1 Graphical Analysis of Datasets:

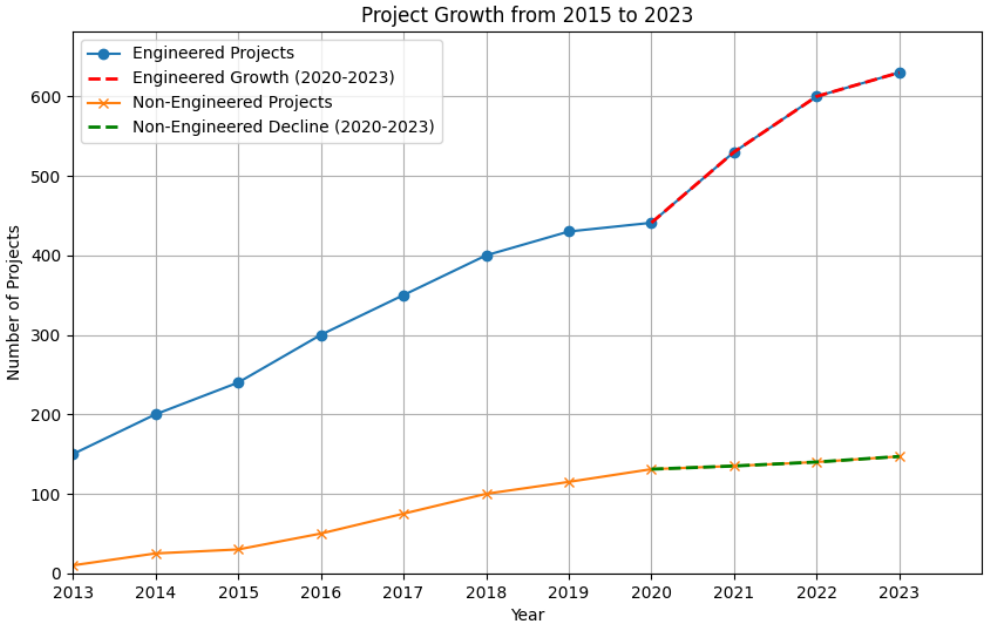

Figure 4.2: Graph Analysis Of Datasets

This graph illustrates the evolution of the NICHE dataset, capturing the growth in the number of engineered and non-engineered machine learning projects from 2015 to 2023. With the latest update in 2023, we observe a noteworthy rise in the proportion of engineered projects, which are defined by the adoption of robust software engineering principles, as opposed to non-engineered projects.

In the 2023 update, the NICHE dataset was augmented with an additional 60 projects, categorized as engineered machine learning projects, and 15 projects as non-engineered. This update underscores the ongoing effort to refine the dataset, focusing on projects that embody higher software engineering standards to ensure improved accuracy and quality for research purposes.

The trend clearly demonstrates a strategic emphasis on engineered projects, which grew significantly in the period from 2020 to 2023, as indicated by the red dashed line. This selective growth policy aligns with the overarching goal of enhancing the dataset with high-quality, engineered machine learning projects that are more likely to provide valuable insights for the MSR and MLR communities. Concurrently, the

number of non-engineered projects remained relatively stable, with a slight decline noted, as shown by the green dashed line.
The proactive curation of the dataset in 2023, which led to the identification and inclusion of 60 new engineered projects, reflects a deliberate shift towards enriching the dataset with contributions that are not only numerous but also of substantial quality. This approach ensures that the NICHE dataset remains an authoritative resource for researchers, contributing to a more nuanced understanding of software engineering practices within the rapidly evolving field of machine learning.

4.2 Evaluation Datasets accuracy by using J48 Classifier:

This section evaluates the quality of the NICHE dataset and its subsequent update, NICHE-23, by employing the J48 decision tree classifier within the Weka software environment [11]. Objective was to assess the classifier's performance in distinguishing between engineered and non-engineered machine learning projects before and after the 2023 dataset update.
The J48 classifier, an implementation of the algorithm in Weka, was trained separately on the original NICHE dataset and the updated NICHE-23 dataset. The performance metrics of interest were the accuracy, precision, recall, and F-measure, which are standard indicators of a classifier's effectiveness.

Classifier output

```
Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances         443               77.4476 %
Incorrectly Classified Instances       129               22.5524 %
Kappa statistic                          0
Mean absolute error                      0.3493
Root mean squared error                  0.4179
Relative absolute error                 99.8499 %
Root relative squared error             99.9997 %
Total Number of Instances              572

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.500     0.226     N
                 1.000    1.000    0.774      1.000    0.873      ?        0.500     0.774     Y
Weighted Avg.    0.774    0.774    ?          0.774    ?          ?        0.500     0.651
```

Figure 4.3: Classifier Result of NICHE dataset

In the figure above, Upon training the J48 classifier with the outdated NICHE dataset, the classifier achieved an accuracy of 77 percent in correctly classifying instances, with a weighted average precision of 0.7. These metrics serve as a baseline for the dataset's ability to enable accurate predictive modeling.

Classifier output

```
Time taken to build model: 0.01 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances         510               81.0811 %
Incorrectly Classified Instances       119               18.9189 %
Kappa statistic                          0.335
Mean absolute error                      0.3035
Root mean squared error                  0.3896
Relative absolute error                 84.6331 %
Root relative squared error             92.0538 %
Total Number of Instances              629

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.299     0.033     0.733      0.299     0.425      0.383    0.648     0.411     N
               0.967     0.701     0.819      0.967     0.887      0.383    0.648     0.824     Y
Weighted Avg.  0.811     0.545     0.799      0.811     0.779      0.383    0.648     0.727
```

Figure 4.4: Classifier Result of NICHE-23 dataset

In contrast, the updated NICHE-23 has shown a marked improvement in classification performance. The J48 classifier trained on NICHE-23 achieved an accuracy of 81 percent for correctly classified instances and a weighted average precision of 0.8. This enhancement in classifier performance can be attributed to the more rigorous curation and categorization of projects in the updated dataset, reflecting the increased delineation between engineered and non-engineered projects.
The results indicate that the refined project selection criteria and the updated project metrics in NICHE-23 have contributed to a more accurate and reliable foundation for classification models. This improved accuracy is essential for researchers utilizing the dataset to develop tools for automated quality assessment and to conduct empirical research in machine learning and software engineering.

The NICHE-23 dataset's augmentation offer profound insights into the evolving landscape of ML project development. The distinction between engineered and non-engineered projects, based on comprehensive engineering practices, provides a valuable framework for analyzing the quality and sustainability of ML software. This dataset not only facilitates the empirical study of ML projects' software engineering aspects but also serves as a benchmark for developing classifiers and other research

tools. The update significantly impacts the overall structure of the NICHE dataset by ensuring a precise representation of high-quality projects, enabling researchers to adapt to and reflect emerging trends. This continuous evolution of the dataset underscores its importance for both Mining Software Repository (MSR) and Machine Learning Research (MLR), offering a dynamic resource for exploring best practices, fostering innovation, and improving project outcomes in the ML field.

4.3 Scientific Contributions and advancements

Methodological Impact Our work introduces innovative methodologies for curating and updating the NICHE-23 dataset, setting a new standard in dataset management within machine learning and software engineering research. We developed a comprehensive selection and evaluation framework for machine learning projects by filter them with most widely used libraries (PyTorch and TensorFlow [12]), incorporating advanced tools and algorithms for data analysis, such as the use of cloc10 for accurate line-of-code counts. This meticulous approach ensures the dataset's ongoing relevance and utility, offering a replicable model for future dataset curation efforts.

Utility for MSR and MLR Research The NICHE-23 dataset significantly advances research in mining software repositories (MSR) and mining machine learning repositories (MLR), facilitating a broad spectrum of studies. It enables comparative analyses of ML projects, exploration of project evolution, and detailed studies on the application of SE practices in ML settings. By providing a curated, comprehensive data source, our dataset accelerates the exploration and understanding of ML and SE practices, supporting the development of tools for automated quality assessment and empirical research.

Future Research Directions The contributions of this research also lay the groundwork for future research directions and have potential educational implications. By identifying gaps and opportunities for further study, this dataset encourages ongoing exploration of ML and SE practices. Additionally, it can be used as a teaching resource to illustrate real-world applications of ML and SE principles, thereby contributing to the education of future researchers and practitioners.

CHAPTER 5

Conclusion and Future Work

The update to the NICHE-23 dataset significantly enriches the landscape of research into machine learning (ML) project practices by meticulously categorizing projects into "Engineered" and "Non-engineered" based on defined engineering criteria. This enhancement not only provides a clearer lens through which to view the quality and sustainability of ML software projects but also establishes a benchmark for evaluating the integration of software engineering practices.

This comprehensive dataset are instrumental for guiding future research directions, including the development of tools for automated quality assessment, conducting comparative analyses across different ML domains, and exploring the educational potential of these findings in shaping future ML developers. Moreover, the ongoing expansion and refinement of the NICHE dataset will be crucial in adapting to technological advancements and emerging project practices, ensuring its continued relevance and utility in fostering a deeper understanding of successful machine learning project development.

Bibliography

[1] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" in *Proceedings of the Fourth International Workshop on Mining Software Repositories*, May 2007. (Cited at page 1)

[2] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promise and perils of mining github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. Association for Computing Machinery, 2014, pp. 92–101. [Online]. Available: https://doi.org/10.1145/2597073.2597074 (Cited at page 1)

[3] 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*. Association for Computing Machinery, 2020, pp. 431–442. [Online]. Available: https://doi.org/10.1145/3379597.3387473 (Cited at page 2)

[4] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, 2017. [Online]. Available: https://doi.org/10.1007/s10664-017-9512-6 (Cited at pages 3, 4, 8, 12 and 13)

[5] P. Pickerill, H. J. Jungen, M. Ochodek, M. Mackowiak, and M. Staron, "Phantom: curating github for engineered software projects using time-series clustering," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2897–2929, 2020. [Online]. Available: https://doi.org/10.1007/s10664-020-09825-8 (Cited at pages 3 and 12)

[6] D. Gonzalez, T. Zimmermann, and N. Nagappan, "The state of the mluniverse: 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, pp. 431–442, [Online]. Available: https://doi.org/10.1145/3379597.3387473. (Cited at page 3)

[7] R. Widyasari and et al., "Niche: A curated dataset of engineered machine learning projects in python," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 2023, pp. 62–66. [Online]. Available: https://arxiv.org/pdf/2303.06286 (Cited at pages 3, 4, 5, 7, 11 and 12)

[8] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, "Does code decay? assessing the evidence from change management data," *Software Engineering, IEEE Transactions on*, vol. 27, no. 1, pp. 1–12, 2001. (Cited at page 13)

[9] 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. (Cited at page 15)

[10] H. Nguyen, F. Lomio, F. Pecorelli, and V. Lenarduzzi, "Pandora: Continuous mining software repository and dataset generation," *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 263–267, 2022. (Cited at page 15)

[11] G. Holmes, A. Donkin, and I. H. Witten, "Weka: a machine learning workbench," in *Proceedings of ANZIIS '94 - Australian New Zealand Intelligent Information Systems Conference*, 1994, pp. 357–361. (Cited at page 19)

[12] 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*. Association for Computing Machinery, 2022, pp. 1–11. [Online]. Available: https://doi.org/10.1145/3524610.3527897 (Cited at page 22)