Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 387–392
DOI: 10.15439/2025F3421
ISSN 2300-5963 ACSIS, Vol. 43

Comparative Analysis of AI Software and Traditional Software GitHub Repositories Using Process Mining

Oguzhan Tasci, Tugba Gurgen Erdogan
0009-0009-2549-758X, 0000-0003-1491-8739
Hacettepe University, Department of Computer Engineering
Beytepe Campus, 06800 Ankara/Türkiye
Email: oguzhantasci5561@gmail.com, tugba@cs.hacettepe.edu.tr

***Abstract*—This study conducts a comparative analysis of Artificial Intelligence (AI) software and traditional GitHub repositories, focusing on workflow efficiency and sentiment dynamics. Using process mining and sentiment analysis techniques, we examine repositories from eight prominent projects, encompassing diverse datasets of issues and pull requests filtered for relevance and consistency. Our findings reveal that AI software repositories exhibit different workflow patterns and sentiment dynamics compared to traditional repositories. Sentiment analysis uncovers that contributors to AI software repositories experience more positive sentiment dynamics, likely reflecting structured workflows and collaborative tools. Conversely, traditional repositories exhibit longer resolution times and more fluctuating sentiment patterns, which may indicate higher complexity or less automation. These insights provide valuable recommendations for optimizing repository management, fostering contributor satisfaction, and improving collaborative software development environments.**

I. Introduction

MANAGING software development processes that are multi-disciplinary, human-intensive, and prone to change requires effective tools. Git, a free and open-source distributed version control system, is widely used to manage projects of all sizes with speed and efficiency. GitHub, an open-source hosting platform, extends Git's capabilities by offering cloud-based version control and collaboration features. As one of the leading data sources for software development research [1], GitHub enables developers to manage and store their code while providing a wealth of information for analyzing software development processes, particularly in collaborative software projects.

As artificial intelligence (AI) has become integral to modern software systems, understanding the development processes of AI software systems has emerged as a significant area of research in software engineering [2]. GitHub provides rich data, including information on issues, commits, projects, users, comments, and the defined GitHub development flow.

Process mining is a data science technique that extracts process insights from event logs stored in information systems [3]. GitHub data, generated during the software development can be converted to event logs to enable process discovery and analysis. By identifying elements of the actual software development process, such as issues, pull requests (PR), and commits, along with their descriptive attributes, process deviations and inefficiencies can be systematically investigated.

Mining Software Repositories (MSR) techniques focus on extracting and analyzing data from software repositories to uncover interesting, useful, and actionable insights about software systems. MSR is considered one of the fastest-growing and most compelling areas in the field of software engineering [4]. The literature contains numerous studies focused on mining GitHub repositories to analyze software development processes for specific purposes. These include detecting anomalies in PR processes [5], understanding and improving students' project workflows [6], [7], [8], managing issues in AI software repositories [9], and analyzing the effects of continuous integration and continuous deployment (CI/CD) practices on repositories [10]. Several tools such as GitHubDataViz [11], GitLab Analyser [6], and Git Truck [12] have been introduced to support the analysis of software development data by visualizing Git issue and commit data.

In this study, we present a GitHub-based data collection framework for process mining and an empirical analysis comparing actual software development processes using process mining techniques and sentiment dynamics in AI software and traditional software repositories. Specifically, this research explores whether AI software projects exhibit distinct patterns in issue resolution, PR management, and contributor sentiment compared to traditional repositories.

II. Related Work and Background

Poncin et al. [13] demonstrated the application of process mining in software repositories, facilitating the analysis of developer roles and bug report handling. Expanding on this, Macak et al. [7] used process mining techniques for the Git log of students' projects to understand their development processes.

Sentiment analysis has become an important tool in understanding the emotional undertones in software development processes. Comments of repository artifacts, such as issues, PRs, and code, contain unstructured and valuable data to understand the sentiment dynamics that impact workflow efficiency and collaboration.

Jurado and Rodriguez [14] conducted a study that explored the use of sentiment analysis to monitor distributed teams, highlighting how emotional analysis of developer-written issues and tickets can provide a richer understanding of the development environment. Yang et al. [15] carried out an empirical study specifically focused on sentiment analysis in the GitHub repositories, revealing that the emotional tone of comments significantly affects the speed of bug fixing. Guzman et al. [16] identified patterns indicating higher negative sentiment in Java-based projects and more positive sentiment in geographically distributed teams. Research into developer behavior in open-source projects indicates a strong correlation between developer activities and sentiment shifts, which can serve as predictors of project success [17]. Robinson et al. [17] provided insights into how these factors influence repository health and productivity. Sinha et al. [18] also analyzed the sentiment in commit logs, exploring the emotional undertones in commits and their impact on project outcomes. While these studies offer valuable perspectives on developer sentiment, our research differentiates itself by investigating how the development processes of AI software products might influence these sentiments.

Zhou Yang et al. [9]'s empirical study on issue management in AI software repositories highlighted challenges such as runtime errors and unclear instructions.

Our study extends previous studies to a broader range of repository artifacts, including issues, code comments, and commit messages, to provide a more holistic view of sentiment dynamics in software development processes, which are discovered through process mining techniques.

III. Methodology

A. Repository Selection and Data Collection

Data was collected from publicly available GitHub repositories using the GitHub REST API. AI software repositories (TensorFlow, PyTorch, scikit-learn, Keras) focus on machine learning algorithms and frameworks. Traditional repositories (Angular, React, Bootstrap, Node.js) emphasize conventional software technologies. The selected repositories are listed in Table I.

TABLE I: Selected Repositories for Analysis

Repository	Type	Stars	Issues	PRs
TensorFlow [19]	AI software	187k	37k	31.1k
PyTorch [20]	AI software	86.1k	31.1k	85.7k
Scikit-Learn [21]	AI software	60.8k	9.9k	16.9k
Keras [22]	AI software	62.4k	11.9k	7.3k
Node.js [23]	Traditional	109k	16.1k	33.4k
Bootstrap [24]	Traditional	171k	23.9k	15.1k
Facebook-React [25]	Traditional	232k	10.4k	15.5k
Angular [26]	Traditional	96.7k	27.9k	25.2k

Selection criteria included high repository activity, star count indicating popularity, contributor diversity, and clear domain classification.

B. Data Preparation

Issues, PRs, and comments from the repositories were extracted. For issues, the title, body, state (open/closed), timestamps (created, updated, closed), and associated labels were collected. For PRs, the title, body, state, timestamps, and merge details were gathered. Additionally, comments associated with both issues and PRs were extracted and used for sentiment analysis.

The data collected from the repositories underwent several data preprocessing stages to ensure its quality and suitability for analysis. These steps included data cleaning, data filtering, data formatting, data balancing, and data validation techniques.

Records with incomplete or missing critical information were either supplemented with placeholder text or discarded. Duplicate entries were identified and removed to ensure the uniqueness of each record.

Text processing techniques such as tokenization, stopword removal, and lemmatization were applied to the comment data. A custom `clean_text` function was used to standardize text data by converting to lowercase and removing unnecessary characters such as punctuation marks and numbers.

Several filters were applied to ensure the dataset remained focused and relevant. Records with fewer than five comments were removed to focus on meaningful discussions. Issues or PRs that had not been updated in the last year or were created before 2020 were removed to ensure current relevance. Only merged or closed PRs and issues were retained for further analysis. Extreme outliers with unusually high or low comment counts were removed to prevent skewing results.

Dates and times were standardized to ISO 8601 format. Additional features such as time-to-close for issues and PRs were created. Categorical variables were encoded numerically for machine learning models.

To ensure fair representation of both AI software and traditional repositories, the dataset was carefully balanced using a stratified approach. Datasets from each category were combined in proportional measures, maintaining equal representation of issues and PRs across both repository types.

Data validation steps included spot checks using functions like `sample()` and `head()`, statistical summaries of key attributes, and missing data analysis using the k-nearest neighbors (KNN) imputation for numerical values and mode imputation for categorical values.

The dataset was enhanced by integrating process mining data, structured into event logs containing key attributes such as case ID, activity, timestamp, and resource attributes. These logs were validated using process mining tools like Disco [27] and PM4Py [28] to ensure they met the necessary quality standards for analysis.

C. Data Modelling

The data model for this research integrates various data entities, each representing key elements of the GitHub repository workflow. Figure 1 illustrates the interconnections among users, repositories, issues, PRs, commits, and comments.

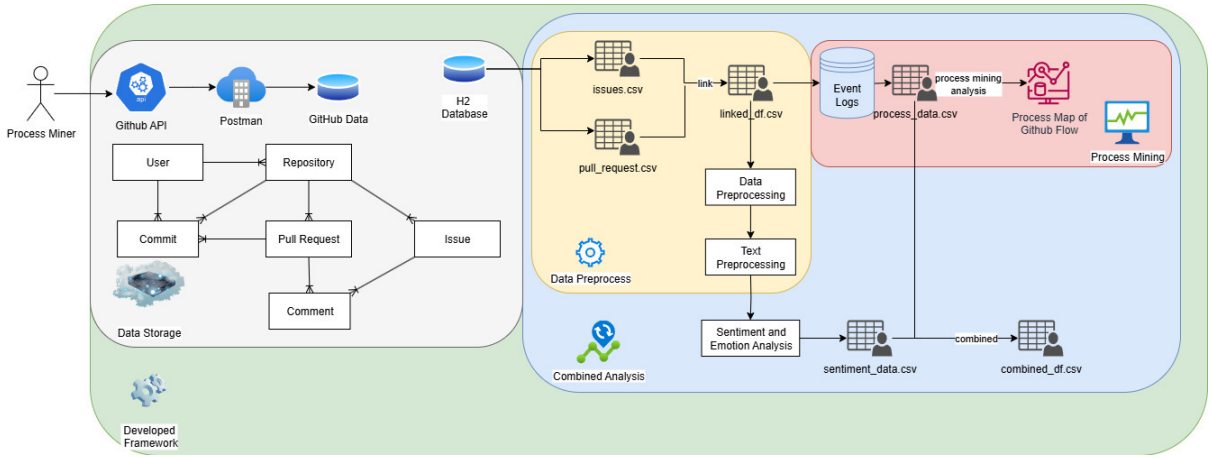


Fig. 1: Overview of the Developed Framework for GitHub Workflow Analysis

The core entities in the data model include users, repositories, issues, PRs, commits, and comments. A user can own or contribute to multiple repositories and can create or be assigned to various issues. Similarly, users can create or review multiple PRs, as well as produce numerous commits and comments.

Each repository in the model can contain multiple issues, PRs, and commits. Issues are typically associated with multiple comments, forming a one-to-many relationship. Similarly, PRs can encompass multiple commits and comments, with each commit associated with a specific PR.

Moreover, comments are crucial for communication during the development process. They are linked to either an issue or PR, serving as feedback or further discussion on a specific task. These interactions enable developers to refine code, resolve issues, and collaborate effectively.

The data is collected using the GitHub API, with retrieval performed via Postman. The collected data is stored in an H2 Database [29]. Subsequently, the raw data is transformed into two files: issues.csv and pullrequest.csv. These files are integrated into a third file, linkeddf.csv, which links issues to their corresponding PRs for analysis. This integration is crucial for understanding the relationship between issues and PRs and enables the analysis of their lifecycle within the repository.

The dataset is subjected to extensive data cleaning and text processing to prepare it for sentiment and emotion analysis. Additionally, the processed data is analyzed using process mining techniques to generate visual process maps, helping to trace the flow of actions such as issue or PR creation, updates, closures, and their relationship with sentiment.

D. Predictive Process Modeling

The structured dataset formed the basis for training various machine learning models aimed at analyzing the workflow in GitHub repositories. The predictive models developed in this study focused on two primary tasks: classifying issues and PRs, and predicting sentiment and emotional tone from text data. This section outlines the steps followed in the modeling process, which include feature engineering, model selection, and model training.

1) Feature Engineering: The first step in predictive modeling was feature engineering, where key attributes from the 'issues.csv' and 'pull_request.csv' files were extracted. The features considered for classification include text-based attributes such as the length of issue descriptions, the titles of PRs, and the presence of specific keywords that may indicate the nature of the task (e.g., "bug", "feature", and "documentation"). These features are crucial for distinguishing between different types of issues and PRs.

Additionally, sentiment-related features were generated from the text data. Sentiment scores (positive, negative, neutral) were extracted using sentiment analysis tools, which helped gauge the emotional tone of the text.

2) Predictive Model Selection and Training: A variety of machine learning models were considered to classify issues and PRs into categories such as "bug", "feature request", or "documentation update". Models such as Random Forest, Support Vector Machines (SVM), XGBoost, Naive Bayes, and Logistic Regression were selected for their proven effectiveness in text classification tasks.

Sentiment analysis was performed on comments within issues and PRs using the Valence Aware Dictionary and sEntiment Reasoner (VADER) [30] model in this study. VADER is well-suited for evaluating sentiment patterns, as it is specifically designed to detect the intensity of sentiments expressed in social media-like content. The analysis categorized comments as positive, neutral, or negative, with aggregated results providing insights into sentiment trends throughout the issue resolution and PR processes.

To complement sentiment analysis, emotion analysis was conducted using a pre-trained Bidirectional Encoder Representations from Transformers (BERT) [31] model, which had been fine-tuned for emotion classification. BERT was chosen due to its ability to capture subtle contextual nuances in text, making it particularly effective for analyzing the emotional tone of developer comments. The model identified several

emotions, including neutral, sadness, surprise, anger, joy, fear, and disgust.

VADER and BERT have limitations with domain-specific technical language and context length variations common in software development discussions.

These emotional insights were then correlated with key metrics such as issue resolution times and overall repository activities, aiming to explore the potential impact of developer emotions on workflow efficiency.

The final step in the modeling phase involved deploying the best-performing models in a real-time environment. These were integrated into an automated pipeline that classifies incoming issues and PRs, predicts their sentiment, and performs process mining analysis.

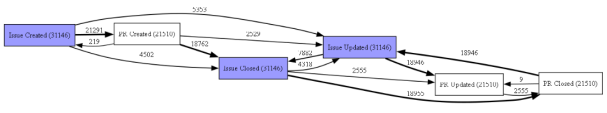
IV. Results and Discussion

A. Process Discovery of Software Repositories

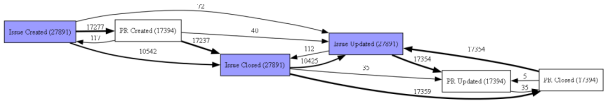
Event logs of repositories are used to discover the process map of the repositories using DFG Miner of the PM4Py library [28]. Figure 2 illustrates the process maps of PyTorch (AI software repository) and Angular (traditional software repository), showing common states including issue created, issue updated, PR created, issue closed, PR updated, and PR closed.

The flow of issues typically follows a consistent pattern: creation, updates, and eventual closure. However, certain issues are directly closed without further updates, while others may lead to the creation of corresponding PRs before they are closed. Similarly, PRs generally undergo creation, updates, and closure, although some are updated shortly before closure.

Notably, Angular exhibits a higher frequency of issues and PR interactions, indicating a more rapid and iterative development approach, which may be typical for traditional software development practices. In contrast, PyTorch demonstrates fewer PR updates and fewer instances where PRs are generated following issues, suggesting a more cautious and deliberate workflow, potentially attributed to the complexity and specificity of AI-related code. These findings emphasize key differences in workflow efficiency and developer behavior between AI software and traditional repositories.



(a) PyTorch Process Map



(b) Angular Process Map

Fig. 2: Process Maps for AI software and Traditional Repositories

B. Sentiment Analysis of the Repositories

Figure 3 depicts the distribution of sentiments across varying time-to-close durations for issues. AI software repositories exhibit a greater density of issues with neutral to positive sentiment and shorter resolution times. In contrast, traditional repositories exhibit a wider spread and with more cases of longer time-to-close, regardless of sentiment.

The data shows that issues with positive or neutral sentiment are generally resolved more faster in both types of repositories, possibly due to their less critical or complex nature. However, traditional repositories have a more distributed model, suggesting variability in the handling and resolution of issues, which may reflect different workflows or resource allocations.

Fig. 3: Comparison of Issue Resolution Times

Figure 4 illustrates the distribution of sentiment scores. We see that AI software repositories have a significant peak around the neutral sentiment (0), with a lower positive sentiment peak, suggesting that issues in AI repositories often maintain a balanced or positive view, even with their complexity. In traditional repositories, there is a larger spread towards positive sentiments, potentially reflecting trust in established development practices and community support.

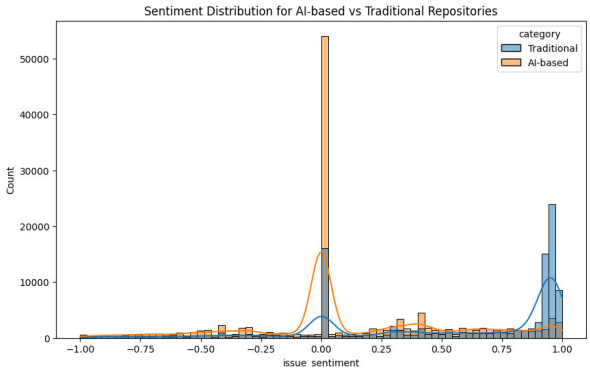


Fig. 4: Sentiment Polarity Distribution

C. Emotion Analysis of the Repositories

Figure 5 provides insight into the common emotional responses in AI software and traditional repositories. In particular, "sadness" and "fear" emotions are high in both categories.

However, AI software repositories show a significantly higher frequency of "sadness" compared to traditional repositories. Also, "anger" and "surprise" are observed more in AI software repositories, while traditional repositories show slightly higher levels of "fear".

This distribution suggests that problems in AI software repositories may trigger stronger negative emotions. This is probably due to the complexity and rapid evolution characteristic of AI projects. Contributors may experience frustration and even hesitation when working with advanced models and algorithms, which may increase emotional reactions. Traditional repositories, in contrast, seem to reflect a more balanced emotional distribution, possibly because they are dealing with well-established technologies and often face problems with familiar, predictable solutions.

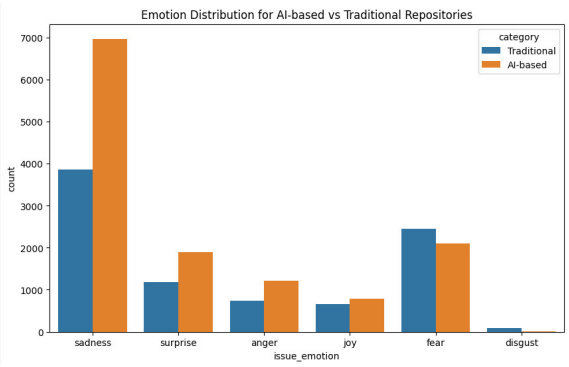


Fig. 5: Emotion Distribution for AI software vs. Traditional Repositories

D. Comparison of Issue Labels the Repositories

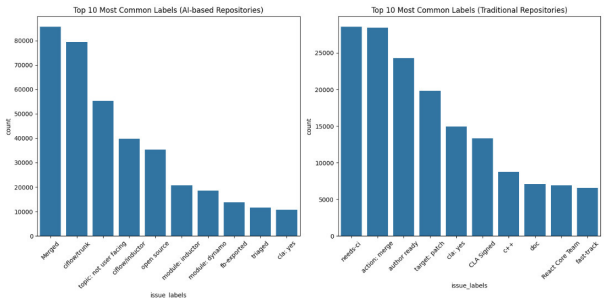
Figure 6a displays the top 10 most common issue labels in and Figure 6b also displays the top 10 labels with the highest average sentiment for both AI software repositories (left) and traditional repositories (right).

Figure 6 compares issue labels. AI software repositories frequently use "Merged," "ciflow/trunk," emphasizing automated testing crucial in ML projects. Traditional repositories show "Needs-ci," "action: merge," emphasizing formal CI processes.

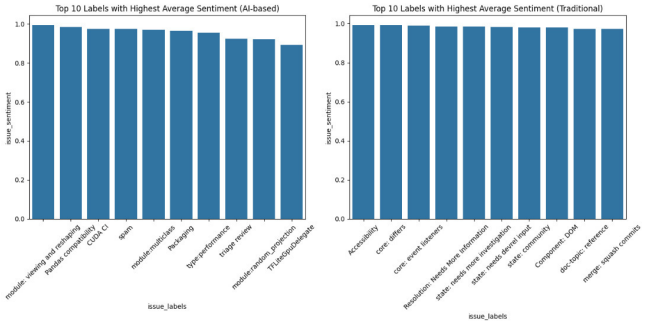
The comparison reveals that while both AI software and traditional repositories focus on merging contributions and continuous integration, AI software repositories have a greater emphasis on modularity and rapid integration of changes, while traditional repositories follow more structured, incremental workflows. These distinctions align with the experimental nature of AI development versus the stability and predictability required in traditional software engineering.

High-sentiment labels differ: AI software repositories focus on performance optimization ("CUDA CI," "type-performance"), while traditional repositories emphasize core functionality ("Accessibility," "core: event listeners").

The comparison reveals that while both AI software and traditional repositories foster positive sentiment toward issue resolution, their focus areas differ. AI software repositories emphasize modularity, performance optimization, and compatibility across frameworks, which are crucial in cutting-edge AI/ML development. In contrast, traditional repositories concentrate on core stability, accessibility, and detailed issue investigation, which aligns with the needs of more mature, stable software projects.



(a) Most Common Labels



(b) Highest Sentiment Labels

Fig. 6: Label Analysis in AI software and Traditional Repositories

E. Resolution Time Comparison

We compare the resolution time of issues and PRs for each repository type and list them in Table II and III. AI software repositories take longer to close issues (55.04 vs 43.63 days) but close PRs faster (36.68 vs 40.43 days). Both show similar median times (2-3 days), indicating most items are resolved quickly regardless of domain.

TABLE II: Time to Close Comparison for Issues

Category	Mean (days)	Median (days)
AI software	55.04	2
Traditional	43.63	2

TABLE III: Time to Close Comparison for Pull Requests

Category	Mean (days)	Median (days)
AI software	36.68	3
Traditional	40.43	3

V. Conclusion

This study provides a comparative analysis of AI software and traditional GitHub repositories, focusing on workflow efficiency and sentiment dynamics.

A comparison between AI software repositories, such as PyTorch, and traditional repositories, such as Angular, highlighted significant differences in development practices. The process maps indicated that workflows in AI software repositories are often more deliberate and cautious, with fewer PR updates, while traditional repositories tend to exhibit more iterative processes with frequent PR updates. These findings are instrumental for assessing workflow efficiency and serve as a foundation for exploring how sentiment and emotion analysis might impact productivity and developer behavior within these repositories.

These findings demonstrate how software domain characteristics influence development patterns, workflow efficiency, and contributor sentiment.

Despite these promising results, this study has several limitations. The dataset is limited to eight repositories due to GitHub REST API rate limits (5,000 requests/hour), potentially restricting the generalizability of the findings. Additionally, the analysis does not account for code coverage of AI code of the repositories and variations in team size, project complexity, or contributor demographics, which may influence repository performance.

Future research could address these limitations by exploring a broader range of repositories and incorporating additional contextual factors. Furthermore, integrating explainable AI methods to assess issue complexity dynamically could provide deeper insights into the decision-making processes of contributors and administrators.

VI. Acknowledgment

This work was supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

References

[1] João Caldeira, Fernando Brito e Abreu, Jorge Cardoso, Rachel Simões, Toacy Oliveira, and José Pereira dos Reis. Software development analytics in practice: A systematic literature review. *Archives of Computational Methods in Engineering*, 30(3):2041–2080, 2023.

[2] Tugba Gurgen Erdogan, Haluk Altunel, and Ayça Kolukısa Tarhan. A process model for ai-enabled software development: A synthesis from validation studies in white literature. *Journal of Software: Evolution and Process*, page e2743, 2024.

[3] Wil Van Der Aalst and Wil van der Aalst. *Data science in action*. Springer, 2016.

[4] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. Pydriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 908–911, 2018.

[5] Bohan Liu, He Zhang, Weigang Ma, Hongyu Kuang, Yi Yang, Jinwei Xu, Shan Gao, and Jian Gao. Mining pull requests to detect process anomalies in open source software development. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.

[6] Bjoern M Eskofier. Exploration of process mining opportunities in educational software engineering-the gitlab analyser. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 601–604, 2020.

[7] Martin Macak, Daniela Kruzelova, Stanislav Chren, and Barbora Buhnova. Using process mining for git log analysis of projects in a software development course. *Education and information technologies*, 26(5):5939–5969, 2021.

[8] Saimir Bala Thanh Nguyen and Jan Mendling. Multi-dimensional process analysis of software development projects. pages 179 – 186, 2024.

[9] Zhou Yang, Chenyu Wang, Jieke Shi, Thong Hoang, Pavneet Kochhar, Qinghua Lu, Zhenchang Xing, and David Lo. What do users ask in open-source ai repositories? an empirical study of github issues. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 79–91. IEEE, 2023.

[10] Jeffrey Fairbanks, Akshharaa Tharigonda, and Nasir U Eisty. Analyzing the effects of ci/cd on open source repositories in github and gitlab. In *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 176–181. IEEE, 2023.

[11] Rifat Ara Proma and Paul Rosen. Visual analysis of github issues to gain insights. *arXiv preprint arXiv:2407.20900*, 2024.

[12] K Højelse, T Kilbak, J Røssum, E Jäpelt, Leonel Merino, and M Lungu. Git-truck: Hierarchy-oriented visualization of git repository evolution. In *2022 Working Conference on Software Visualization (VISSOFT)*, pages 131–140. IEEE, 2022.

[13] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Process mining software repositories. In *2011 15th European Conference on Software Maintenance and Reengineering*, pages 5–14. IEEE, 2011.

[14] Francisco Jurado and Pablo Rodriguez. Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues. *Journal of Systems and Software*, 104:82–89, Jun. 2015.

[15] Bo Yang, Xinjie Wei, and Chao Liu. Sentiments analysis in github repositories: An empirical study. In *2016 Asia-Pacific Software Engineering Conference Workshops (APSECW)*, pages 67–74. IEEE, 2016.

[16] Eleni Guzman, David Azócar, and Yijun Li. Sentiment analysis of commit comments in github: An empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, pages 352–355, 2014.

[17] William N. Robinson, Tianjie Deng, and Zirun Qi. Developer behavior and sentiment from data mining open source repositories. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 5386–5395. IEEE, 2016.

[18] Vinayak Sinha, Alina Lazar, and Bonita Sharif. Analyzing developer sentiment in commit logs. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 520–523. IEEE, 2016.

[19] TensorFlow Repository. An open source machine learning framework for everyone, 2025. Last accessed 1 January 2025.

[20] PyTorch. Tensors and dynamic neural networks in python with strong gpu acceleration, 2025. Last accessed 1 January 2025.

[21] Scikit-Learn. scikit-learn: machine learning in python, 2025. Last accessed 1 January 2025.

[22] Keras Team. Deep learning for humans, 2025. Last accessed 1 January 2025.

[23] Node.js. Node.js is an open-source, cross-platform javascript runtime environment, 2025. Last accessed 1 January 2025.

[24] Bootstrap. The most popular html, css, and javascript framework for developing responsive, mobile first projects on the web, 2025. Last accessed 1 January 2025.

[25] Facebook React. The library for web and native user interfaces, 2025. Last accessed 1 January 2025.

[26] Angular. Deliver web apps with confidence, 2025. Last accessed 1 January 2025.

[27] Christian W Günther and Anne Rozinat. Disco: Discover your processes. In *Demonstration Track of the 10th International Conference on Business Process Management, BPM Demos 2012*, pages 40–44. CEUR-WS. org, 2012.

[28] Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. Pm4py: A process mining library for python. *Software Impacts*, 17:100556, 2023.

[29] Thomas Mueller. *H2 Database Engine*, 2023. Version 2.2.222, last accessed 27 May 2025.

[30] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media (ICWSM-14)*, pages 216–225. AAAI, 2014.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019. Fine-tuning methodology described within the original BERT paper.