International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Navigating Technical Debt in the Evolving Landscape of Machine Learning and Artificial Intelligence

Sowmya Ramesh Kumar

Seattle, WA
Email: *rsowmyash[at]gmail.com*

Abstract: ***This paper illuminates the profound impact of Machine Learning (ML) and Artificial Intelligence (AI) in modern industries, charting their journey from academic concepts to vital business tools. The surge in implementation, however, introduces a critical challenge: technical debt. Expanding on three key dimensions, code dependencies, data dependencies, and system dependencies—the paper explores the entanglement challenge in code, untangling complexity in data dependencies, and addressing high-debt design patterns in systems. It advocates for a holistic approach to measuring and mitigating technical debt, emphasizing guiding questions and good practices. The article concludes by stressing the imperative of fostering a culture that recognizes and prioritizes the reduction of technical debt for sustained success in the dynamic landscape of ML and AI.***

Keywords: Machine learning, artificial intelligence, refactoring, hidden feedback loops, technical debt, artificial intelligence, coding, production.

1. Introduction

Machine Learning (ML) and Artificial Intelligence (AI) have become pivotal players in the technological landscape, witnessing an unprecedented surge in adoption across diverse industries. The last few years have seen these technologies evolve from theoretical constructs tested on academic datasets to indispensable tools underpinning critical business operations. This paradigm shift, however, is not without its challenges, and one of the most formidable adversaries is the accrual of technical debt.

The meteoric rise of ML and AI has led to a frenetic pace of implementation in production environments. Organizations, eager to capitalize on the potential business impact, often prioritize speed over meticulous engineering. Consequently, technical debt, a concept introduced by Ward Cunningham, has become a significant concern in the realm of ML and AI projects. Much like financial debt, technical debt is incurred for the sake of expediency and building new systems, but its unbridled accumulation can have detrimental effects on the overall health of these projects.

In this dynamic landscape, where research in ML and its sibling field, Deep Learning, is expanding exponentially, managing technical debt is a critical imperative. As organizations transition from proof - of - concept stages to integrating ML and AI into actual business processes, the challenges associated with technical debt manifest in three primary aspects [1]: code dependencies, data dependencies, and system dependencies.

Code Dependencies: The Entanglement Challenge

In traditional software engineering, best practices often advocate for the establishment of robust abstraction boundaries through encapsulation and modular design [2]. These principles contribute to creating maintainable code, facilitating isolated changes, and enabling incremental improvements. However, the application of these practices becomes inherently challenging in ML systems, where adherence to specific intended behavior is elusive.

The entanglement challenge arises as ML models, designed as tools for amalgamating data sources, create intricate dependencies. This phenomenon, encapsulated in the CACE (pronounced as "cake") Principle—Changing Anything Changes Everything—underlines the difficulty in isolating improvements. The entanglement resulting from ML packages can make the isolation of changes practically impossible, leading to a rapid accrual of technical debt.

Addressing this challenge involves a dual - pronged approach. Firstly, isolating models and service ensembles is crucial, striking a balance between managing separate models and enforcing modularity to mitigate entanglement. Secondly, gaining deep insights into the behavior of model predictions becomes paramount. This can be achieved through the development of visualization tools and metrics spanning multiple dimensions, providing a nuanced understanding of the system's intricacies.

Another facet of code dependencies is the presence of hidden feedback loops [3]. Systems that learn from real - world behavior, such as predicting click - through rates of news headlines, often rely on user clicks as training labels. This creates a feedback loop where predictions from the model influence user behavior, leading to challenges in analyzing system performance. Detecting and eliminating hidden feedback loops are essential steps in mitigating technical debt in this aspect.

Data Dependencies: Untangling Complexity

Data dependencies in ML models introduce a layer of complexity that is distinct from code dependencies [3]. These dependencies can stem from various sources, including input features from other ML models, and can be harder to identify and untangle. Legacy features, bundled features, and epsilon

(ϱ) - features contribute to the entanglement of data dependencies.

Legacy features, once integral to a model, may become redundant over time as new features are added. Detecting this redundancy and removing such features is crucial to reducing data dependencies. Similarly, bundled features, added under time constraints, can mask the value of individual features, leading to unnecessary complexity. Regularly evaluating and testing features can help identify those that contribute little or no value.

ML researchers, driven by the pursuit of improved accuracy, may introduce epsilon (ϱ) - features to a model even when the gain in accuracy is marginal. This practice, while seemingly beneficial, adds complexity overhead. Mitigating this involves periodically evaluating the impact of individual features on model accuracy and removing those that do not significantly contribute.

System Dependencies: Tackling Design Patterns

System dependencies in ML projects often result in high - debt design patterns. Glue code and pipeline jungles are two prominent anti - design patterns that pose challenges and should be avoided or refactored.

Glue code, often employed by ML researchers to develop general - purpose solutions, becomes unwieldy as it accumulates outside the core packages. While these generic systems are useful, managing the supporting code can become onerous and costly for long - term maintenance. Reducing glue code involves re - implementing specific algorithms within the broader system architecture, thereby minimizing the need for extensive supporting code.

Pipeline jungles, a subtype of glue code, commonly emerge in data preparation. These complex webs of scrapes, joins, and sampling steps can evolve organically, making them challenging to manage and prone to errors. Redesigning data collection and feature extraction processes holistically can help avoid the pitfalls of pipeline jungles, even though it requires a significant investment of engineering effort.

Measuring and Paying Off Technical Debt: A Holistic Approach

While the metaphor of technical debt resonates strongly in the software engineering realm, it lacks strict metrics for tracking over time. Balancing the need for teams to move quickly with the imperative of establishing low - debt good practices is an ongoing challenge [5]. Several questions can guide teams in assessing and addressing technical debt:

Does improving one model or signal degrade others?
How quickly can new team members be onboard with the existing codebase and architecture?
How easily can a new algorithm be tested at full scale?

In addition to these considerations, adopting good practices is essential. This includes code refactoring, enhancing unit tests, erasing dead code and comments, reducing dependencies, and regularly reviewing documentation. These practices contribute to the overall health of ML and AI projects by ensuring that technical debt is managed and mitigated effectively.

Paying ML - related technical debt requires dedicated commitment and a shift in mindset and team culture. Recognizing, prioritizing, and rewarding efforts aimed at reducing technical debt should be integral pillars in fostering the long - term health and success of data science teams. This commitment ensures that as ML and AI technologies continue to evolve, the foundations supporting their implementation remain robust and resilient.

2. Conclusion

Effectively managing technical debt in ML and AI projects requires addressing three critical dimensions: code dependencies, data dependencies, and system dependencies. Strategies involve isolating models and gaining insights into their behavior, untangling complexity in data dependencies, and reducing high - debt design patterns in systems.

A holistic approach to measuring and mitigating technical debt involves posing crucial questions and adopting good practices such as code refactoring and regular documentation reviews. Success in paying down ML - related technical debt hinges on a committed team culture that recognizes, and rewards efforts aimed at reducing technical debt. By systematically addressing these challenges, data science teams ensure the enduring health and success of ML and AI implementations amidst ongoing technological advancements.

References

[1] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D.,. . . & Young, M. (2014). Machine learning: The high interest credit card of technical debt.
[2] Sculley, D. et al. “Hidden Technical Debt in Machine Learning Systems. ” *Neural Information Processing Systems* (2015).
[3] Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2016). What’s your ML test score? A rubric for ML production systems.
[4] A. Zheng. The challenges of building machine learning tools for the masses. SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)
[5] M. Fowler. Refactoring: improving the design of existing code. Pearson Education India, 1999.