This paper is a preliminary version and the final version will be published at the 3rd IDE Workshop co-located with ICSE 2026.

Proto-ML: An IDE for ML Solution Prototyping

Selin Coban
RWTH Aachen University
Research Group Software Construction
Aachen, Germany
coban@swc.rwth-aachen.de

Miguel Perez
RWTH Aachen University
Research Group Software Construction
Aachen, Germany
miguel.perez@rwth-aachen.de

Horst Lichter
RWTH Aachen University
Research Group Software Construction
Aachen, Germany
lichter@swc.rwth-aachen.de

ABSTRACT

Prototyping plays a critical role in the development of machine learning (ML) solutions, yet existing tools often provide limited support for effective collaboration and knowledge reuse among stakeholders. This paper introduces Proto-ML, an IDE designed to strengthen ML prototyping workflows. By addressing key deficiencies such as insufficient stakeholder involvement, limited cross-project knowledge reuse, and fragmented tool support, Proto-ML offers a unified framework that enables structured documentation of prototyping activities and promotes knowledge sharing across projects.

The Proto-ML IDE consists of three extension bundles: prototype implementation, analysis, and knowledge management. These extensions support tasks ranging from evaluating prototype quality against defined criteria to incorporating stakeholder perspectives throughout the development process. Preliminary user feedback suggests that Proto-ML can increase prototyping efficiency and foster more transparent and reusable ML solution development.

CCS CONCEPTS

• **Software and its engineering → Integrated and visual development environments**.

KEYWORDS

Integrated Development Environment, Machine Learning, Prototyping

ACM Reference Format:
Selin Coban, Miguel Perez, and Horst Lichter. 2026. Proto-ML: An IDE for ML Solution Prototyping. In *Proceedings of (3rd International IDE Workshop).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Prototyping is a long-standing practice in software engineering used to explore design alternatives, reduce uncertainty, and obtain early feedback before full-scale system development. In traditional software prototyping, developers typically rely on established engineering practices, with prototypes focusing on system architecture, user interfaces, or selected functionalities.

In contrast, prototyping of machine learning (ML) solutions is inherently exploratory and highly data-centered. The outcome of an ML prototype depends not only on design choices but also on the availability and quality of data, the selection and configuration of algorithms, the evaluation of model performance, and ultimately the acceptance of the ML prototype among stakeholders.

Current tool support for ML solution prototyping largely builds on tool environments designed for general-purpose software development. Widely used tools such as Jupyter Notebook, version control systems, and documentation platforms are frequently combined in an ad hoc manner. While effective for quick experimentation, this fragmented tool landscape provides limited support for documentation of abandoned paths, traceability of design decisions or reusing knowledge. As a result, knowledge generated during ML prototyping is often lost, and collaboration across disciplinary boundaries is hindered [16, 25].

These observations motivate the need for dedicated tool support tailored to the specific characteristics of ML solution prototyping. In this paper, we present the design and implementation of an IDE for ML solution prototyping (referred to as Proto-ML). This work builds upon and extends our initial results for tool-supported ML solution prototyping, as presented in [1].

2 ML SOLUTION PROTOTYPING

ML solution prototyping is a form of experimental prototyping to quickly evaluate alternative ML models for a given ML problem. The developed *ML prototype* serves as both a demonstrator and an executable artifact, providing a shared, interactive basis for communication between stakeholders and go/no-go decisions.

In this section, we describe essential deficiencies that motivate our work and introduce the ML solution prototyping process.

2.1 Deficiencies

A literature review conducted by Truss et al. [30] pointed out that one of the most significant challenges in ML prototyping is the knowledge disparity among involved stakeholders. Further studies highlight that non-technical stakeholders often lack access to key information necessary for a general understanding of ML prototypes [6, 15, 24, 34]. They may not know how the ML prototype works, what data and ML models are used, or how trustworthy the results are. In addition, stakeholders often lack insight into why certain design decisions were made. We discuss these deficiencies in the area of knowledge reuse and the provision of information to non-technical stakeholders in detail below.

3rd International IDE Workshop, ,

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

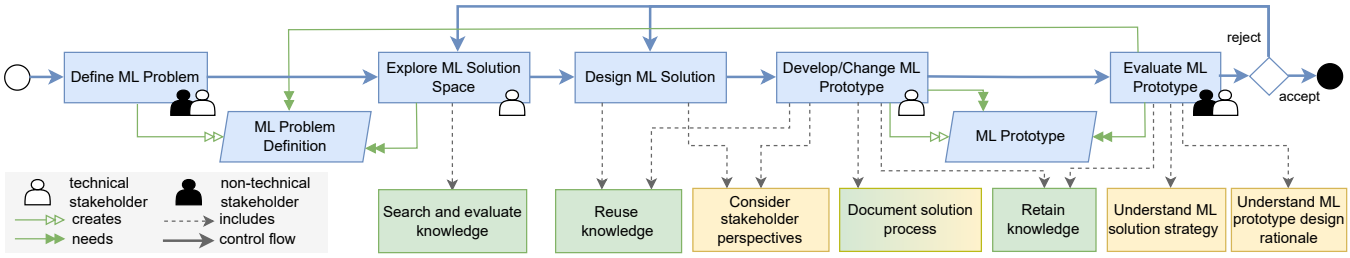

Figure 1: The ML solution prototyping process (adapted from [1])

Knowledge is not reused systematically: The knowledge created during the prototyping of an ML solution encompasses all artifacts and insights generated during that process. This covers tangible outputs, such as software components, ML problem-definition documents, and evaluation reports. It also encompasses intangible assets like experiences, lessons learned about data characteristics, and assessments of different solution strategies. These experimental insights are particularly valuable because they can inform future prototyping activities. For example, data scientists may be able to exclude specific approaches from the outset.

We conducted an interview study [7] to investigate how reuse is practiced in ML solution prototyping, uncover existing reuse practices, and analyze how tools support these activities, as tools play a key role in accelerating the retrieval and application of reusable artifacts. We observed that developers systematically *search for* and *evaluate knowledge* prior to *reuse the knowledge* and *retain the knowledge*. Notably, the scope of reuse extends beyond code artifacts to encompass ML algorithms, solution design patterns, and step-by-step procedural instructions.

For effective reuse of knowledge developers must *document the prototyping process* and systematically *retain the knowledge* gained to make it available for future prototyping endeavors. Unfortunately, current tool environments do not adequately support the entire knowledge reuse management process. Instead, developers must spend considerable time searching for knowledge, evaluating it, and storing it appropriately.

Stakeholder-specific information is not provided: ML prototypes are evaluated using a range of quality criteria, which may be quantitative (e.g., accuracy and precision) or qualitative (e.g., fairness and legal compliance). The active involvement of all stakeholders is essential because each brings unique perspectives and distinct concerns regarding the ML problem [5]. Domain experts, in particular, can provide valuable insights into whether the ML prototype will be accepted within its intended application area. It is therefore crucial to *consider stakeholder perspectives* throughout the entire ML solution prototyping process. Stakeholders must receive sufficient and comprehensible information so they can *understand the ML solution strategy* and *understand the design rationale* behind specific design decisions.

2.2 The ML Solution Prototyping Process

Figure 1 depicts the basic ML solution prototyping process in blue. A central artifact is the *ML problem definition*, which specifies the scope, business objectives, and success criteria. Non-technical stakeholders (e.g., domain and business experts) primarily influence the design of this artifact. Guided by the ML problem definition, technical stakeholders (e.g., developers and data scientists) conduct the subsequent activities: *exploring the ML solution space*, *designing the ML solution*, and *developing and changing the ML prototype*. Domain and business experts participate in *evaluating the ML prototype*. Acceptance decisions depend on contextual understanding, anticipated business impact, and ethical and regulatory compliance.

To address the deficiencies in ML solution prototyping described above, we have extended the basic prototyping process to include missing activities that are needed to overcome those deficiencies. On the one hand, these integrate the systematic reuse of knowledge into the prototyping process (shown in green). On the other hand, the extended process includes activities (shown in yellow) that are necessary to better involve non-technical stakeholders, especially in the evaluation of the ML prototype.

3 TOOL REQUIREMENTS

For the proposed ML solution prototyping process to be implemented efficiently, it must be supported by suitable tools.

Based on the tasks performed in the activities, we formulate requirements for the tool environment aimed at increasing efficiency in the reuse process and addressing the stakeholder information needs. A high degree of automation is required to foster broader acceptance of the solution. We have therefore established the following tool requirements for the newly introduced process activities:

- **Search and evaluate knowledge & Retain knowledge**
  R1: *Provide means to manage knowledge sources, capture their evaluation results, and trace them to the ML prototype.*
- **Reuse knowledge**
  R2: *Support fast retrieval of relevant knowledge.*
- **Consider stakeholder perspectives**
  R3: *Automatically evaluate ML prototype against defined stakeholder interests.*
- **Develop/Change ML prototype**
  R4: *Automatically assess ML prototype against defined quality criteria.*
- **Understand ML solution strategy**
  R5: *Provide automated generation of high-level prototype solution overviews or summaries.*
  R6: *Provide automated generation of ML prototype documentation tailored to stakeholders.*

- **Understand ML prototype design rationale**
  R7: *Provide means to explore the prototyping process.*
- **Document solution process**
  R8: *Automatically record the prototyping process and design decisions.*

R1–R2 address improving cross-project knowledge reuse. They require mechanisms to manage knowledge sources, support developers in evaluating them, and enable traceability between reused knowledge and the components in the ML prototype to which it corresponds. In addition, since developers currently spend considerable time searching for relevant knowledge, dedicated support for faster retrieval is needed.

R3 and R4 aim to support developers during ML prototype development by checking it against quality attributes and stakeholder interests. This is expected to result in a higher-quality prototype.

To enhance stakeholder understanding of the ML prototype and its prototyping process, R5–R7 focus on the tailored presentation of the prototype's solution strategy and its prototyping process.

Finally, R7 is a prerequisite for R8. Without recording the prototyping process, it is not possible to provide information about the process.

Note that this list of requirements is not exhaustive and may be expanded in the future.

4 COMPONENTS OF THE PROTO-ML IDE

All existing tools for prototyping ML solutions and those to be developed based on these requirements should be made available in a consistent prototyping IDE, which we have named Proto-ML.

Jupyter Notebooks, often combined with IDEs such as PyCharm or Visual Studio Code, are the de facto standard for developing ML prototypes [31, 32]. For this reason, we have designed the Proto-ML IDE so that Jupyter Notebooks are its central component, with additional bundles of interoperable prototyping-specific tooling extensions. Based on the requirements outlined in Section 3, we have developed three task-oriented extension packages with a total of seven tools to support ML prototyping. Figure 2 provides an overview of the bundles and extensions of the Proto-ML IDE, along with references to the supported activities and the tool requirements presented. In addition to these bundles, the Proto-ML IDE contains a *repository* in which all data relating to current and previous ML prototypes is stored.

We implemented proof-of-concept prototypes[1] using the JupyterLab platform for almost all extensions and evaluated them initially. In the following, we present a first glance at these bundles. Note that some extensions are still under development.

4.1 Bundle: Analysis

To better understand the solution strategy implemented in an ML prototype, we suggest two extensions.

Explainer: The Explainer provides a high-level overview of the activity-flow, data-flows between activities, and basic explanations. This helps stakeholders understand the general structure of the ML prototype. To obtain a high-level view of a notebook, each cell represents a semantic activity. By applying heuristics over inter-cell data dependencies, activity-flow diagrams can be derived. Activity names and concise textual descriptions are automatically generated by LLMs. Finally, the Explainer provides a visualization that enables exploration of both the activities and the corresponding data flow. A screenshot of an activity-flow diagram for a sample notebook is provided in Figure 3.

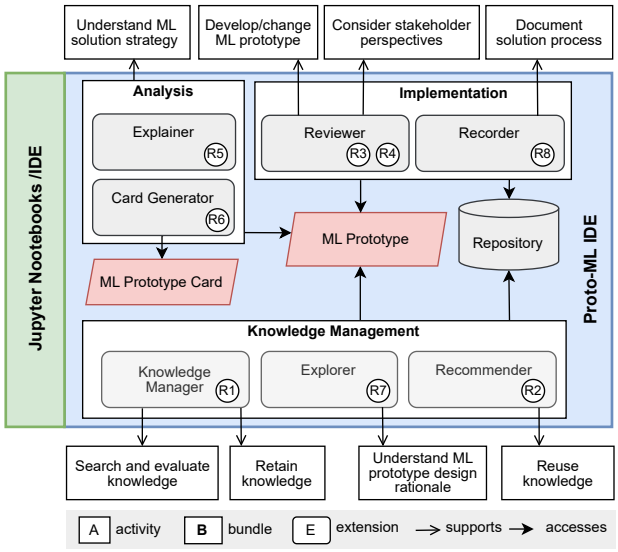

Figure 2: Components of the Proto-ML IDE

We collected initial feedback from five ML practitioners with varying technical backgrounds. All participants reported that the Explainer helped them understand given notebooks more quickly and navigate their own notebooks during development. One participant also indicated their intention to use the Explainer to ensure the notebook reflects their intended implementation.

Card Generator: Further details about an ML prototype are stored in an *ML prototype card* inspired by model cards [14], an established approach for documenting essential information about ML models. We aim to adapt this concept for ML prototypes and tailor it the information needs of technical and non-technical stakeholders. The Card Generator creates these ML prototype cards semi-automatically. This extension is currently under development.

4.2 Bundle: Implementation

In this bundle, we introduce two extensions designed to support developers in creating high-quality ML prototypes that incorporate stakeholder perspectives, as well as an additional extension aimed at automatically documenting the prototyping process.

Reviewer: Similar to static code quality analyzers, we include a Reviewer that evaluates the current state of the ML prototype against ML prototype-specific quality criteria (see Figure 4). To this end, we developed three artifacts that provide these criteria: (1) a *quality model*, (2) a *review checklist*, and (3) *stakeholder personas*. The quality model was synthesized by combining established quality models for ML systems [27, 28] and best-practice guidance for notebooks [18].

[1] url: https://selincoban95.github.io/proto-ml.github.io/

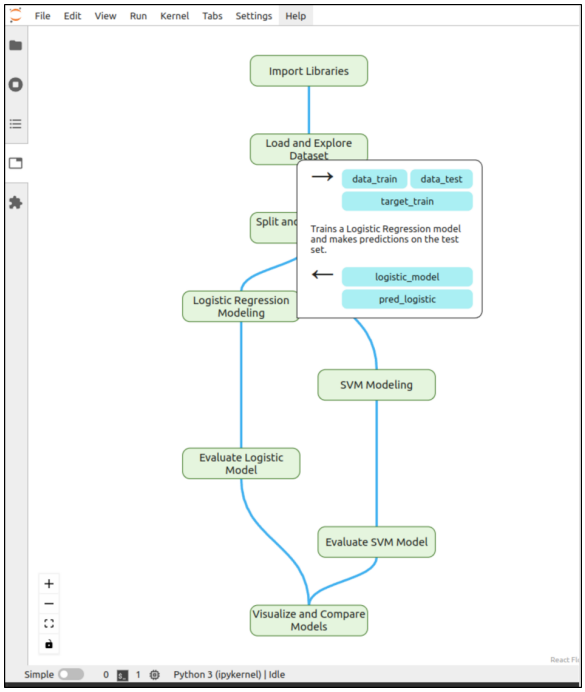

Figure 3: Screenshot of the Explainer extension

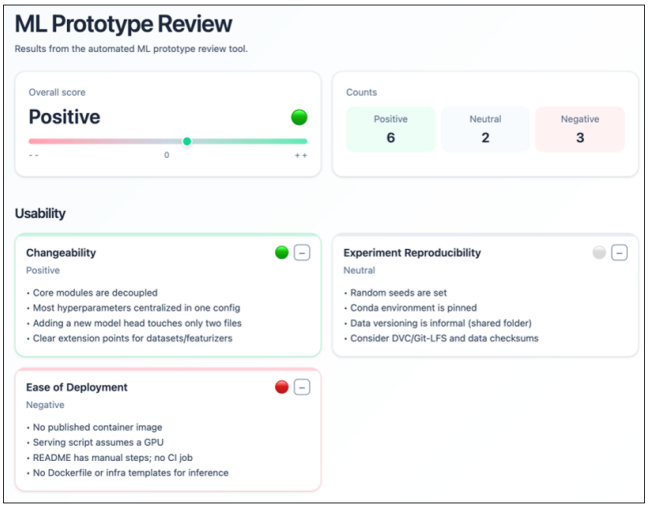

Figure 4: Screenshot of the Reviewer extension dashboard

The review checklist contains items to be considered in notebooks (e.g., include a brief description of the ML problem at the top of the notebook), drawing inspiration from the ML prototype card. To incorporate stakeholder perspectives, we designed *stakeholder personas*. For each persona, we identified the specific requirements of stakeholders based on their role and intentions. Together, these artifacts enable an automated, systematic, and stakeholder-oriented evaluation of the ML prototype.

Recorder: To document the prototyping process, we include a Recorder that automatically tracks all prototyping steps performed in the notebook by storing snapshots. Storing a snapshot of the notebook after every minor modification would quickly lead to excessive amounts of stored data. To address this, we capture snapshots only for semantically meaningful modifications by treating the execution of a single cell as an atomic activity, such as data processing or model training. The stored snapshots enable time-travel across different versions of the ML prototype. While linear version traversal is already supported by existing tools such as Git or Verdant [10], we introduced the concept of an *experiment tree* [29]: When a developer navigates to an earlier version of the notebook and modifies it, a new branch is created in the version history. This branching mechanism allows developers to capture alternative solution paths in a manner consistent with the exploratory nature of rapid experimentation.

4.3 Bundle: Knowledge Management

Since existing tools do not yet support knowledge reuse in ML solution prototyping, we include extensions to support searching, evaluating, reusing, and retaining knowledge.

Explorer: The Explorer presents data from the recorded prototyping process to a stakeholder and allows exploration of past prototyping endeavors, along with comments on design decisions. Using the experiment tree generated by the Recorder, the Explorer provides a tree-based visualization of the notebook's evolution (see Figure 5). Each node in the tree represents a snapshot of the notebook at a specific point in time. When a developer selects a node, the corresponding version is opened. If the developer modifies this version, a new branch is automatically created in the experiment tree. By hovering over a node, additional contextual information becomes available, such as a diff view highlighting changes and any developer comments associated with that snapshot. This design allows developers to rapidly switch between experiment variants and inspect what has already been tried, within a notebook. More details on our Explorer implementation, along with the results of user experiments, are published in [29].

Recommender: Because notebook code cells are the most frequently reused artifact across notebooks, we developed a recommendation system that operates at two granularity levels: (i) notebook-level, suggesting notebooks similar to the one currently open; and (ii) cell-level, suggesting cells similar to the developer's active cell (see Figure 6). The Recommender leverages a vector database populated with representations derived from personal notebooks and a corpus of Kaggle notebooks [20] to enable content-based retrieval. Although offline processing (e.g., training and indexing) can be time-consuming, online inference is low-latency, delivering recommendations interactively during editing.

More details on the Recommender and its performance evaluation can be found in [2]. Applying standard recommender system metrics, the Recommender consistently achieved strong performance. The analysis also identified characteristics that influence recommendation quality, guiding on when and how the Recommender is most effective and informing future design and optimization.

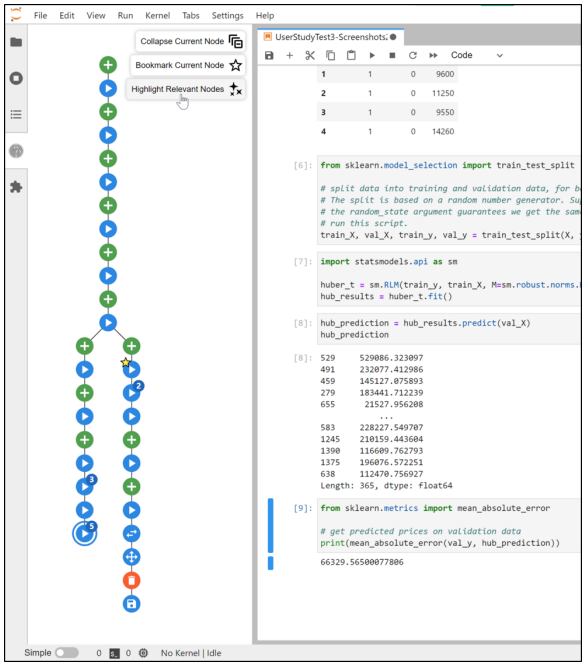

Figure 5: Screenshot of the Explorer extension

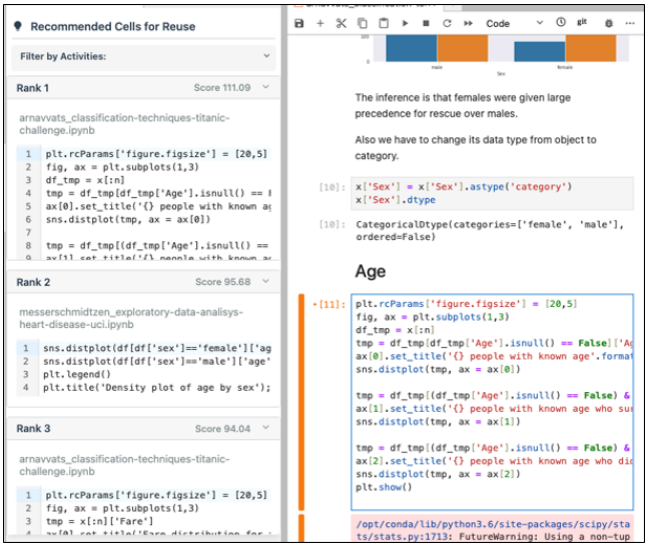

Figure 6: Screenshot of the Recommender extension

Knowledge Manager: Using the Knowledge Manager, developers can store and retain various knowledge sources during exploration of the solution space and trace parts of the ML prototype to the corresponding knowledge sources. Furthermore, the Knowledge Manager includes mechanisms to automatically evaluate the suitability of a knowledge source based on criteria identified in an interview study [7]. For example, it analyzes the source and provides information on whether code is available, whether the author is known and reputable, and whether standard benchmarks were used. As this extension is still in the design phase, further technical and functional details cannot be disclosed at this stage.

5 RELATED WORK

A variety of tools and techniques have been proposed to support isolated aspects of ML solution prototyping, such as code reuse or debugging. A comprehensive review of these techniques and tools around the Jupyter Notebook environment is provided in [26]. In the following, we present some notable tools in the areas of ML prototype analysis, implementation, and knowledge management, structured according to the extension bundles presented in this paper.

Analysis. Bhat et al. [3] introduce DocML, a tool that integrates model cards directly into notebook environments. It assists developers by reminding them of incomplete model card entries and enabling the specification of relevant information through special HTML comments embedded within the corresponding code cells. However, an empirical study by Bracamonte et al. [4] found that the information presented in model cards is often perceived as overly technical, potentially excluding non-technical stakeholders. To address this gap, our Card Generator tailors model cards to the prototyping context by aligning their content with the specific information needs of different stakeholders.

The idea of using visual non-linear workflow models to support stakeholders in understanding ML prototypes was also explored by Ramasamy et al. [23]. The authors evaluated the idea with a visual concept and demonstrated its potential effectiveness in improving comprehension. However, the presented idea was never implemented and evaluated in practice.

Implementation. To support structured development in notebooks, best practices have been proposed [19], and a static notebook analyzer Pynblint has been introduced by Quarnata et al. to automatically check compliance with a subset of these practices aimed at improving collaboration [21]. These best practices are also integrated into the ML prototype's quality model within the Reviewer.

Recording a linear history of notebook versions is supported by the tools presented in [9, 17]. The Recorder builds on the features of Verdant [9] and extends them to support the recording of tree-based histories.

Knowledge Management. Several tools assist developers in retrieving relevant code from past notebooks. One approach is by providing better search support, e.g., through semantic search [11, 12] or complex graph-based queries [8], which both require manual effort from the developer. Existing automatic recommendation approaches are either only tailored to specific tasks, such as data analysis [13, 33], or consider only existing markdown cells in the notebook [22].

6 CONCLUSION AND FUTURE WORK

In this paper, we introduce Proto-ML, an IDE for ML solution prototyping that addresses key deficiencies such as limited stakeholder participation and low reuse potential of prototyping knowledge. We present an enhanced process model for ML solution prototyping that explicitly incorporates activities to mitigate these deficiencies.

From this process model, we derive concrete tool requirements to support the identified activities effectively.

To realize these requirements, we build on the JupyterLab platform and present three extension bundles that support the analysis, implementation and knowledge management of ML prototypes. Our preliminary evaluation of these extensions has received positive feedback from users, indicating their potential to improve prototyping efficiency.

Several extensions of Proto-ML are still under development and need to be completed. Once the full IDE is available, we plan to conduct empirical studies in real-world ML prototyping projects to evaluate its impact on knowledge reuse and stakeholder engagement.

REFERENCES

[1] Selin Aydin and Horst Lichter. 2024. Tool-supported Development of ML Prototypes. In *2024 31st Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 477–481.

[2] Selin Aydin, Dennis Mertens, and Ouyu Xu. 2024. *An Automated Evaluation Approach for Jupyter Notebook Code Cell Recommender Systems*. Universitätsbibliothek der RWTH Aachen.

[3] Avinash Bhat, Austin Coursey, Grace Hu, Sixian Li, Nadia Nahar, Shurui Zhou, Christian Kästner, and Jin LC Guo. 2023. Aspirations and Practice of ML Model Documentation: Moving the Needle With Nudging and Traceability. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[4] Vanessa Bracamonte, Sebastian Pape, Sascha Löbner, and Frederic Tronnier. 2023. Effectiveness and Information Quality Perception of an AI Model Card: A Study Among Non-Experts. In *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*. 1–7. https://doi.org/10.1109/PST58708.2023.10320197

[5] Edgar Brea and Jerad Allen Ford. 2023. Innovative Problem Solving in the Machine Learning Age: Is Domain Expertise Still Important?. In *Academy of Management Proceedings*, Vol. 2023. Academy of Management Briarcliff Manor, NY 10510, 14780.

[6] Andrea Brennen. 2020. What Do People Really Want When They Say They Want "Explainable AI?" We Asked 60 Stakeholders. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–7.

[7] Selin Coban, Patrick Chrestin, and Horst Lichter. 2025. *Uncovering the Knowledge Reuse Process in ML Prototyping based on an Interview Study*. https://owncloud.swc.rwth-aachen.de/s/edtw9xrHknPRfCz (under review).

[8] Misato Horiuchi, Yuya Sasaki, Chuan Xiao, and Makoto Onizuka. 2022. Jupysim: Jupyter Notebook Similarity Search System. In *EDBT*.

[9] Mary Beth Kery, Bonnie E. John, Patrick O'Flaherty, Amber Horvath, and Brad A. Myers. 2019. Towards Effective Foraging by Data Scientists to Find Past Analysis Choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13.

[10] Mary Beth Kery and Brad A. Myers. 2018. Interactions for Untangling Messy History in a Computational Notebook. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 147–155.

[11] Lan Li and Jinpeng Lv. 2024. Unlocking Insights: Semantic Search in Jupyter Notebooks. arXiv:2402.13234 [cs.IR] https://arxiv.org/abs/2402.13234

[12] Xingjun Li, Yuanxin Wang, Hong Wang, Yang Wang, and Jian Zhao. 2021. Nbsearch: Semantic search and visual exploration of computational notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[13] Xingjun Li, Yizhi Zhang, Justin Leung, Chengnian Sun, and Jian Zhao. 2023. EDAssistant: Supporting Exploratory Data Analysis in Computational Notebooks with In Situ Code Search and Recommendation. *ACM Trans. Interact. Intell. Syst.* 13, 1, Article 1 (March 2023), 27 pages.

[14] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 220–229.

[15] Nils Brede Moe, Aybüke Aurum, and Tore Dybå. 2012. Challenges of Shared Decision-Making: A Multiple Case Study of Agile Software Development. *Information and Software Technology* 54, 8 (2012), 853–865. https://doi.org/10.1016/j.infsof.2011.11.006 Special Issue: Voice of the Editorial Board.

[16] Roger Nazir, Alessio Bucaioni, and Patrizio Pelliccione. 2024. Architecting ML-enabled systems: Challenges, Best Practices, and Design Decisions. *Journal of Systems and Software* 207 (2024), 111860. https://doi.org/10.1016/j.jss.2023.111860

[17] Nextjournal GmbH. 2021. The Notebook for Reproducible Research — Nextjournal. https://nextjournal.com/home. [Accessed 13-Jun-2023].

[18] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. 2021. Understanding and Improving the Quality and Reproducibility of Jupyter Notebooks. *Empirical Softw. Engg.* 26, 4 (Jul 2021), 55 pages.

[19] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. 2019. A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (Msr)*. IEEE, 507–517.

[20] Luigi Quaranta, Fabio Calefato, and Filippo Lanubile. 2021. KGTorrent: A Dataset of Python Jupyter Notebooks from Kaggle. *CoRR* abs/2103.10558 (2021). arXiv:2103.10558 https://arxiv.org/abs/2103.10558

[21] Luigi Quaranta, Fabio Calefato, and Filippo Lanubile. 2022. Pynblint: A Static Analyzer for Python Jupyter Notebooks. In *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. 48–49.

[22] Chaiyong Ragkhitwetsagul, Veerakit Prasertpol, Natanon Ritta, Paphon Sae-Wong, Thanapon Noraset, and Morakot Choetkiertikul. 2024. Typhon: Automatic Recommendation of Relevant Code Cells in Jupyter Notebooks.

[23] Dhivyabharathi Ramasamy, Cristina Sarasua, Alberto Bacchelli, and Abraham Bernstein. 2023. Visualising Data Science Workflows To Support Third-Party Notebook Comprehension: An Empirical Study. *Empirical Software Engineering* 28, 3 (2023), 58.

[24] Samar Saeed, Shahrzad Sheikholeslami, Jacob Krüger, and Regina Hebig. 2023. What Data Scientists (Care To) Recall. In *Product-Focused Software Process Improvement - 24th International Conference, PROFES 2023, Dornbirn, Austria, December 10-13, 2023, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 14483)*. Springer, 208–224.

[25] Karthik Shivashankar, Ghadi S. Al Hajj, and Antonio Martini. 2025. Scalability and Maintainability Challenges and Solutions in Machine Learning: Systematic Literature Review. arXiv:2504.11079 [cs.SE] https://arxiv.org/abs/2504.11079

[26] Md Saeed Siddik, Hao Li, and Cor-Paul Bezemer. 2025. A Systematic Literature Review of Software Engineering Research on Jupyter Notebook. arXiv:2504.16180 [cs.SE] https://arxiv.org/abs/2504.16180

[27] Julien Siebert, Lisa Jöckel, Jens Heidrich, Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, and Mikio Aoyama. 2020. Towards Guidelines for Assessing Qualities of Machine Learning Systems. *CoRR* abs/2008.11007 (2020). arXiv:2008.11007 https://arxiv.org/abs/2008.11007

[28] Julien Siebert, Lisa Joeckel, Jens Heidrich, Adam Trendowicz, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, and M. Aoyama. 2022. Construction of a Quality Model for Machine Learning Systems. https://doi.org/10.1007/s11219-021-09557-y

[29] Laurens Studtmann, Selin Aydin, and Horst Lichter. 2023. Histree: A Tree-Based Experiment History Tracking Tool for Jupyter Notebooks. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. 299–308.

[30] Mario Truss and Marc Schmitt. 2025. Human-Centered AI Product Prototyping with No-Code AutoML: Conceptual Framework, Potentials and Limitations. *International Journal of Human–Computer Interaction* 41, 15 (2025), 9304–9319.

[31] Ashwin Prasad Shivarpatna Venkatesh, Samkutty Sabu, Mouli Chekkapalli, Jiawei Wang, Li Li, and Eric Bodden. 2024. Static Analysis Driven Enhancements for Comprehension in Machine Learning Notebooks. arXiv:2301.04419 [cs.SE] https://arxiv.org/abs/2301.04419

[32] Jiawei Wang, Li Li, and Andreas Zeller. 2020. Better Code, Better Sharing: On the Need of Analyzing Jupyter Notebooks. In *Proceedings of the ACM/Ieee 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 53–56.

[33] Alex Watson, Scott Bateman, and Suprio Ray. 2019. PySnippet: Accelerating Exploratory Data Analysis in Jupyter Notebook through Facilitated Access to Example Code. In *EDBT/ICDT Workshops*.

[34] Amy X. Zhang, Michael Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 22 (May 2020), 23 pages.