Tool-supported Development of ML Prototypes

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

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

***Abstract*—Prototyping of machine learning (ML) solutions represents a pivotal stage in developing ML-enabled systems. In this course, the prototype serves as a means of communication and should demonstrate the technical feasibility and value to technical and non-technical stakeholders. But, in the context of the current ML solution prototyping process and tooling environment, non-technical stakeholders are limited in their ability to participate effectively, primarily due to the difficulty of understanding the specific ML solution strategy being implemented.**

In addition, valuable knowledge is lost during the prototype development process because the process is not sufficiently documented, preserved, or made easily accessible for future projects.

To significantly improve the development of ML prototypes, we propose an extended ML prototyping process and tool support in the form of a toolbox. Preliminary implementations of some tools of the toolbox are presented.

***Index Terms*—Machine Learning, Prototyping, Jupyter Notebook**

I. Introduction

The development of software prototypes, in general, has been a well-researched area for some time. However, the prototyping of ML solutions has not been extensively investigated. Moving into product-related development activities prematurely is risky, as clarity on the feasibility and value of the required ML solution is not yet established. Apart from Jupyter Notebooks, which have emerged as the most commonly used tool, there is a notable lack of engineering approaches - particularly methods and dedicated tools - for ML solution prototyping.

Effective communication and decision-making are particularly important during the initial stages of prototyping an ML solution. It is imperative that all involved stakeholders are aligned and on the same page. Utilizing assumptions to address knowledge gaps may, in the long run, result in a more risky development process.

This paper will propose a more systematic approach to ML solution prototyping and tool support in the form of a toolbox.

To this end, Section II establishes the foundations by outlining the prototyping process and introducing relevant terms. The problem statement and the research goals follow in Section III. Our solution, an extended prototyping process and a corresponding toolbox, is proposed in Section IV. Section V describes preliminary results and presents the status of some developed tools. Finally, Section VI places the content in the context of the current research landscape. Section VII concludes the paper and discusses the next steps.

II. Foundations

In the following, we will take a closer look at the concept of prototyping ML solutions.

ML solution prototyping represents a special case of experimental prototyping. Its objective is to evaluate alternative ML models for a given ML problem. The prototyping process follows the CRISP-DM model, which defines the activities of business and data understanding, data preparation, modeling, and evaluation [1].

An *ML prototype* as a result of ML solution prototyping is a functional prototype that implements the production of an ML model by a specific ML workflow and demonstrates the behavior and performance of the ML model. Contentwise, the ML prototype consists mainly of an ML model, the ML workflow that produces/applies it and a performance report. It is usually delivered as a Jupyter Notebook and a custom Python package.

In terms of purpose, an ML prototype serves as both a demonstrator and an executable model. It facilitates interactive experimentation to analyze how the ML solution performs under various modifications or conditions. Consequently, it is used to communicate between stakeholders to decide whether to continue or terminate the project.

Figure 1 illustrates the basic ML solution prototyping process. A key artifact is the *ML Problem Definition*, which documents essential information such as scope, business objectives, and success criteria. This artifact is heavily shaped by non-technical stakeholders such as domain and business experts.

The ML problem definition is the primary input for the following activities, starting with *Exploring the ML Solution Space*, *Implementing the ML Solution*, and *Developing the ML Prototype*. Technical stakeholders, like developers or data scientists, mainly conduct these activities. The preferred tools for the ML prototype development are Jupyter Notebooks, sometimes combined with an IDE.

Domain and business experts are involved in *Evaluating the ML Prototype*. To decide whether to accept or reject the ML prototype, experts must provide deeper insight into the context, the business impact of the ML solution being developed, and ethical and regulatory compliance. It is, therefore, critical that these experts understand the ML prototype. This is also

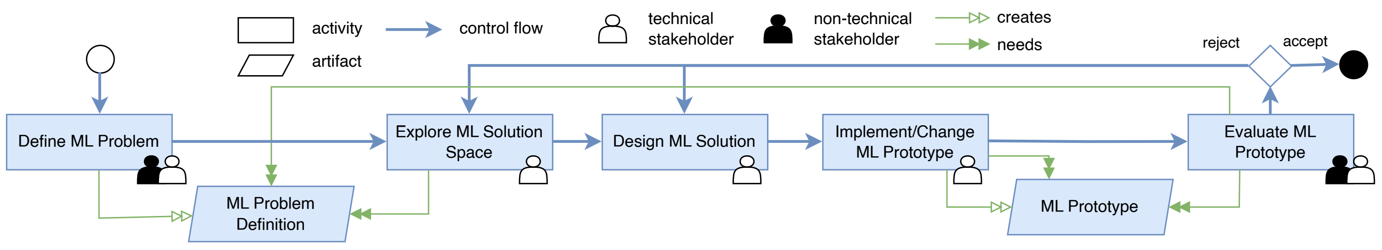


Fig. 1. Basic ML Solution Prototyping Process

essential for building confidence in the ML solution among the domain and business experts, who will ultimately use the ML solution [2] [3].

III. Problem Statement and Goals

To evaluate an ML prototype, it is paramount that all involved stakeholders must have a comprehensive understanding not only of the ”what” (the solution strategy of the ML prototype) but also of the ”why” (the rationale behind design decisions and the reasons for rejecting alternative ML solutions). Regarding the understandability of ML prototypes, recent studies have shown that regardless of the technical expertise, stakeholders need an overview of the key elements of the ML prototype [4] [5], such as the ML algorithms used, the data used for training, validation and test, and the implemented control flow.

The respective information can be collected by examining the source code, but this is a tedious and time-consuming process that requires a certain level of technical expertise. Additional information, e.g., contained in PowerPoint slides, is often incomplete or up-to-date.

During ML solution prototyping, different ML solution approaches are examined and evaluated. Those that prove less promising are discarded, with only the successful ones retained. Furthermore, design decisions leading to the ML prototype are typically not documented, with all implemented ML solutions not saved. This makes it challenging to comprehend how an ML prototype came to be and why a specific ML solution was chosen. The only information source is then the developers of the ML prototype, assuming they remember every step of the ML solution prototyping process.

Furthermore, this valuable information is lost for future ML solution prototyping endeavors. For similar ML problems, it would be beneficial to analyze how these problems were solved in past ML solution prototyping attempts and which ML solution can be excluded from the outset or which ML solution was determined to be very promising.

The analysis of the current situation of ML solution prototyping clearly shows that it must and should be improved. Our research pursues two goals regarding improved ML solution prototyping.

- **Stakeholder Involvement:** In the ML solution prototyping process, all stakeholders should be able to participate regardless of their technical expertise. This allows for a more comprehensive business impact analysis and domain-specific criteria. As it is currently challenging to understand the solution strategy implemented in the ML prototype, non-technical stakeholders, in particular, can only participate in the ML solution prototyping process to a limited extent. This can negatively impact the prototyping process's quality, as assumptions may influence decisions to address knowledge gaps.
- **Cross-Project Reuse:** The knowledge gained through experimentation in the ML solution prototyping process is of value and should be reused. However, this knowledge is not documented, preserved, or made accessible for later reuse.

Both goals can only be achieved if ML prototypes and the ML solution prototyping process are better documented. The lack of available documentation is because it is often not created due to different priorities or a lack of time. Furthermore, manual documentation always requires considerable effort, which is often not planned for.

Therefore, we want to support technical and non-technical stakeholders with an improved prototyping process and tools that automatically extract and capture currently missing information and provide it in a usable format.

IV. ML Solution Prototyping

To achieve the goals described above, we propose an extended ML solution prototyping process and a set of tools to make it more effective and efficient. First, we explain this process; then we present the tools.

A. *ML Solution Prototyping Process*

We enhanced the basic ML solution prototyping process by introducing essential but missing activities and artifacts to achieve the above goals. The extended process is shown in Figure 2.

As stated, up-to-date documentation of the ML prototyping process is crucial. Therefore, we added the *Document ML Prototyping Process* activity, which creates this artifact called *ML Solution Prototyping Documentation*.

To enable non-technical stakeholders to participate actively, especially in evaluating the developed ML prototype, they need to *understand the ML prototype solution strategy* and the *ML prototype design rationale*. Therefore, two respective activities were added to the process.

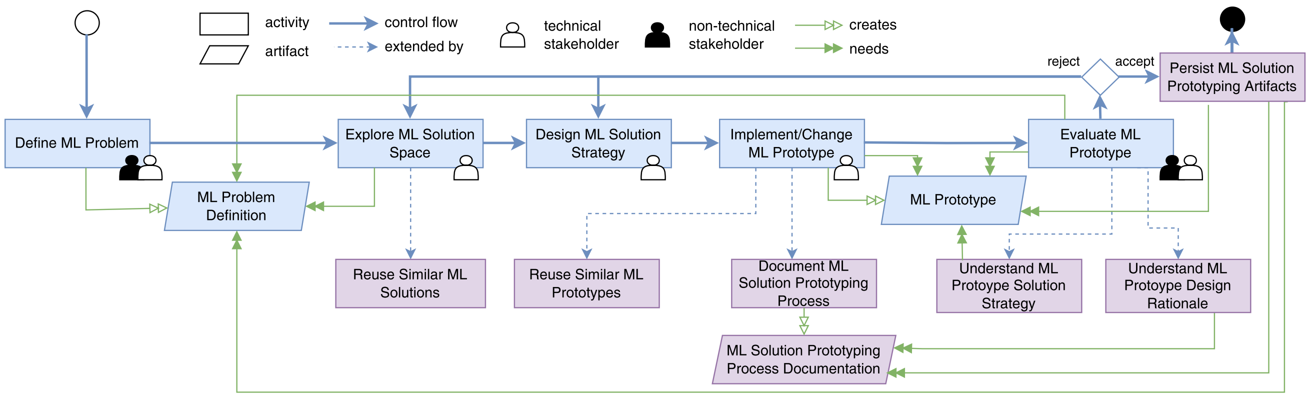

activity
artifact
control flow
extended by
technical stakeholder
non-technical stakeholder
creates
needs
reject
accept
Persist ML Solution Prototyping Artifacts
Define ML Problem
Explore ML Solution Space
Design ML Solution Strategy
Implement/Change ML Prototype
Evaluate ML Prototype
ML Problem Definition
ML Prototype
Reuse Similar ML Solutions
Reuse Similar ML Prototypes
Document ML Solution Prototyping Process
Understand ML Protoype Solution Strategy
Understand ML Protoype Design Rationale
ML Solution Prototyping Process Documentation

Fig. 2. The Extended ML Solution Prototyping Process

Since cross-project reuse is only possible when past ML solution prototyping artifacts are persisted in an easily retrievable manner, we introduced the activity *Persist ML Solution Prototyping Artifacts* as the last activity of the ML solution prototyping process. This enables developers to actively *Reuse Similar ML Solutions* while exploring the ML solution space. Further, they can *Reuse Similar ML Prototypes* during ML prototype development. This would allow for both knowledge and code reuse.

B. ML Solution Prototyping Toolbox

To make the ML solution prototyping process efficient, dedicated tools are needed. To this end, we have conceptually designed the ML solution prototyping toolbox, offering tools to support the stakeholders in performing their activities. The tools and their usage are illustrated in Figure 3.

A common basis for the tools is the *ML SolPt Repository*, which stores all ML problem definitions and their solutions. This includes all versions of the ML solutions and all made design decisions, i.e., the documentation of the corresponding ML solution prototyping process. To support and guide developers in persisting the ML solution prototyping artifacts in this repository, the toolbox provides an *ML PT Persistence Service*.

A central tool for implementing an ML prototype is an *ML Prototype IDE*; often, Jupyter Notebook is used, but other tools have already proven themselves. The ML prototype IDE will be supplemented by the *ML Solution Prototyping Recorder*. It automatically records and stores all prototyping efforts and the corresponding ML prototypes.

The toolbox provides two tools to use the ML solution prototyping repository efficiently. The *ML PT Explorer* allows all stakeholders to actively browse different ML problem definitions and their solutions in an easily accessible way. Since the ML solution prototyping process documentation is stored for each ML prototype, stakeholders can better explore the design rationale for a specific ML prototype. Further, the *ML Prototype Recommender* suggests code from similar ML

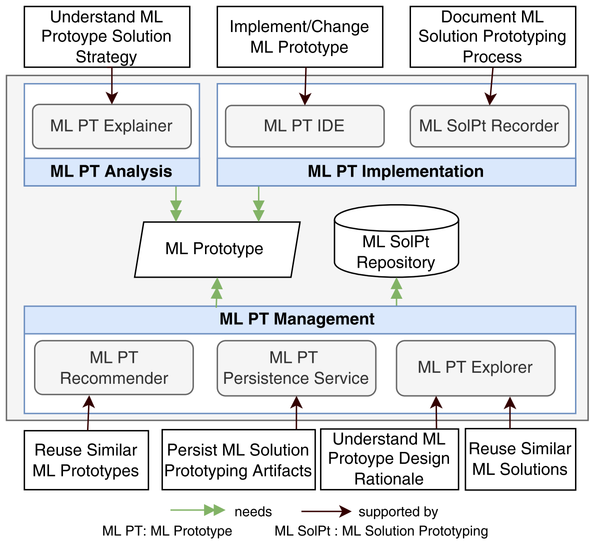

Understand ML Protoype Solution Strategy
Implement/Change ML Prototype
Document ML Solution Prototyping Process
ML PT Explainer
ML PT IDE
ML SolPt Recorder
ML PT Analysis
ML PT Implementation
ML Prototype
ML SolPt Repository
ML PT Management
ML PT Recommender
ML PT Persistence Service
ML PT Explorer
Reuse Similar ML Prototypes
Persist ML Solution Prototyping Artifacts
Understand ML Protoype Design Rationale
Reuse Similar ML Solutions
needs
supported by
ML PT: ML Prototype
ML SolPt : ML Solution Prototyping

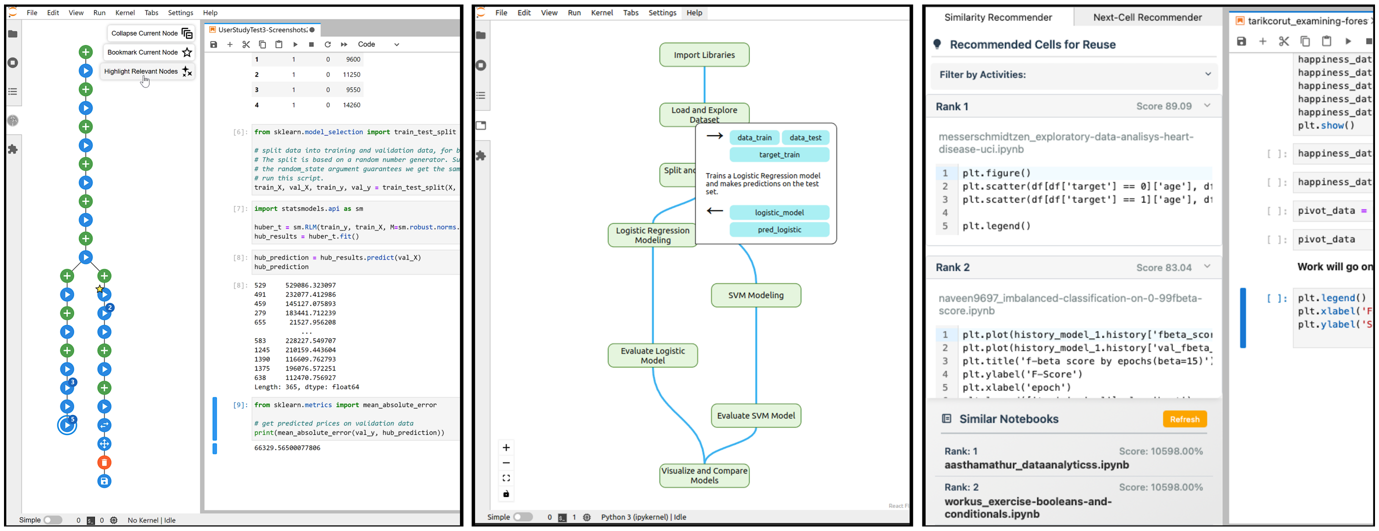
Fig. 3. A Toolbox for ML Solution Prototyping

prototypes or a set of similar ML prototypes to developers during the implementation of the ML prototype.

To simplify the understanding of the ML prototype solution strategy within an ML prototype, the *ML Prototype Explainer* automatically extracts the information relevant to the non-technical stakeholders from the ML prototype and presents it in an easily accessible way, significantly reducing the cognitive complexity of the entire understanding process.

V. Preliminary Results

The research questions have already been partially addressed. Besides a first version of the ML solution prototyping process we developed proof-of-concept implementations for some of the described tools as part of our ML ProtoBox toolbox and collect initial user feedback. Since Jupyter Notebook is the most commonly used tool, we developed the tools



```
1    1    0    9600
2    1    0    11250
3    1    0    9550
4    1    0    14260

[6]: from sklearn.model_selection import train_test_split

     # split data into training and validation data, for b
     # The split is based on a random number generator. Su
     # the random_state argument guarantees we get the sam
     # run this script.
     train_X, val_X, train_y, val_y = train_test_split(X,

[7]: import statsmodels.api as sm

     huber_t = sm.RLM(train_y, train_X, M=sm.robust.norms.
     hub_results = huber_t.fit()

[8]: hub_prediction = hub_results.predict(val_X)
     hub_prediction

[8]: 529     529086.323097
     491     232077.412986
     459     145127.075893
     279     183441.712239
     655      21527.956208
               ...
     583     228227.549707
     1245    210159.443604
     1390    116609.762793
     1375    196076.572251
     638     112470.756927
     Length: 365, dtype: float64

[9]: from sklearn.metrics import mean_absolute_error

     # get predicted prices on validation data
     print(mean_absolute_error(val_y, hub_prediction))

     66329.56500077806
```

messerschmidtzen_exploratory-data-analisys-heart-disease-uci.ipynb

```
plt.figure()
plt.scatter(df[df['target'] == 0]['age'], d
plt.scatter(df[df['target'] == 1]['age'], d

plt.legend()
```

naveen9697_imbalanced-classification-on-0-99fbeta-score.ipynb

```
plt.plot(history_model_1.history['fbeta_sco
plt.plot(history_model_1.history['val_fbeta_
plt.title('f-beta score by epochs(beta=15)'
plt.ylabel('F-Score')
plt.xlabel('epoch')
```

Rank: 1 Score: 10598.00%
aasthamathur_dataanalyticss.ipynb

Rank: 2 Score: 10598.00%
workus_exercise-booleans-and-conditionals.ipynb

```
     happiness_dat
     happiness_dat
     happiness_dat
     happiness_dat
     happiness_dat
     plt.show()

[ ]: happiness_dat

[ ]: happiness_dat

[ ]: pivot_data =

[ ]: pivot_data
```

Work will go on

```
[ ]: plt.legend()
     plt.xlabel('F
     plt.ylabel('S
```

Fig. 4. UI Screenshots of HISTREE, CELLFLOW and JUPYRECSYS

as JupyterLab extensions. In the following section, we will describe the current status of these tools. Screenshot of each tool can be seen in Figure 4.

A. ML PT Explainer: CellFlow

To help stakeholders better understand an ML prototype, we are currently developing a JupyterLab extension named CELLFLOW that extracts, simplifies, and visualizes the control and data flow.

CELLFLOW automatically generates a workflow model that represents and explains the activities performed in the notebook and how artifacts flow between activities. The activities within the workflow model are generated from the given cells, assuming that each cell performs one semantic task, e.g., training of an ML model. Branches in the workflow model are introduced by analyzing the execution dependencies between those cells. We use LLMs to automatically assign meaningful names to activities and to add explanations to activities, as LLMs have been shown to be effective in improving code understandability [6].

The initial feedback from users indicates that the ML solution strategy is understood more rapidly using CELLFLOW.

B. ML SolPT Recorder and PT Explorer: Histree

Data scientists often perform ML solution prototyping through an iterative process, similar to traversing a decision tree. They start by implementing and evaluating one approach, then backtrack to refine certain aspects before re-evaluating. Each approach becomes a unique path within a tree, and branches represent points of divergence for further exploration.

To capture this particular development behavior, we developed a novel tree-based experimentation model for automatically visualizing and documenting the ML solution prototyping process and implemented it in a JupyterLab extension called HISTREE [7]. The extension not only records the execution steps, but also allows users to seamlessly jump between different ML prototype versions and refine them. Early user feedback has been very encouraging.

C. ML PT Recommender: JupyRecSys

What is mainly reused across notebooks during ML prototype development is the implemented code. A study of publicly accessible notebooks on GitHub revealed that 50% of all notebooks lack a unique snippet, consisting exclusively of snippets reused from other notebooks. In particular, code for visualizing data is frequently duplicated [8].

A comprehensive record of past ML solution prototypes should be made easily accessible to developers so that they can reuse the code and knowledge contained in them. This could be achieved by providing a search or recommendation system. Searching requires cognitive effort because you must think about specific keywords or queries to get valuable results.

For this purpose, we have started to build a cell and notebook recommendation system based on similarity to a single cell or an entire notebook, called JUPYRECSYS. This recommender is integrated into JupyterLab as an extension and has access to a vector database containing a large number of notebooks from Kaggle [9]. It can make recommendations as the developer types code into a cell.

An evaluation of the integrated recommendation algorithm shows high precision and accuracy. An extensive usability evaluation will follow.

VI. RELATED WORK

Some initial solution proposals to the aforementioned problems can be found in the fields of software engineering, human-computer interaction, and IDE development.

Numerous best practices have been proposed in various academic papers to enhance the understandability of notebooks, with a particular focus on incorporating comprehensive and robust markdown text [10]. Some papers present tools that automatically generate descriptions for cells [11] [12].

However, one particularly noteworthy contribution in this area is [13], which presents a novel visualization method for non-linear ML workflows and evaluates it in the context of user experiments. The study results indicate that visualizing the workflows helps users gain a significantly more comprehensive understanding of the notebook. The authors only use a static workflow visualization resembling a tree structure for a specific example; a tool that dynamically generates workflows does not yet exist.

Regarding prototyping process recording in Jupyter Notebooks, there is VERDANT [14], a notebook versioning tool that displays a linear history and allows for the inspection of past notebooks. However, it does not permit direct editing or further refinement [15]. Additional tools are either designed to capture linear histories [16] or are incompatible with Jupyter Notebooks [17].

Based on previous notebooks, there are already approaches to make it easier to find and reuse relevant code snippets. However, these tools require manual intervention, such as explicitly storing code snippets in a database for later reuse or creating complex queries to retrieve relevant notebooks [18]. An automated and integrated approach does not yet exist.

Strikingly, there is also very little research on the prototyping process itself, most notably in the lack of consistent terminology.

VII. Conclusions and Future Work

This paper presents prototyping in the context of ML. We identified two problems that mainly affect the understandability of ML prototypes. To improve the development of ML prototypes, an extended ML prototyping process and an ML prototyping toolbox are proposed. Among other features, the toolbox makes it possible to trace the history of an ML prototype, helps to retrieve relevant knowledge from previous ML prototypes, and helps to understand an ML prototype better and faster. Further, we presented the current status of some tool implementations offered by our toolbox ML PROTOBOX.

The subsequent steps in developing ML PROTOBOX will involve implementing the tools in such a way that they function in unison. This will be followed by an extended usability study, which will assess the value of ML PROTOBOX to different stakeholders.

In addition, the basic version of the tools can be improved with more advanced functionalities.

HISTREE is currently focused on capturing Jupyter Notebook operations and versions, organized in the experiment tree model. Future iterations aim to provide a comprehensive historical record of the entire ML solution prototyping process, including operations, code, artifacts, and documentation such as design decisions.

In CELLFLOW, the code within the notebook is the sole focus of the analysis. Since an ML prototype may also contain a custom Python package that is important in the notebook, extending the approach to imported custom Python packages is necessary.

References

[1] C. Shearer, “The CRISP-DM Model: The New Blueprint for Data Mining,” *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[2] A. Brennen, “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*, ser. CHI EA ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–7.

[3] H. Suresh, S. R. Gomez, K. K. Nam, and A. Satyanarayan, “Beyond Expertise and Roles: A Framework to Characterize the Stakeholders of Interpretable Machine Learning and their Needs,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’21. New York, NY, USA: Association for Computing Machinery, 2021.

[4] S. Saeed, S. Sheikholeslami, J. Krüger, and R. Hebig, “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*, ser. Lecture Notes in Computer Science, vol. 14483. Springer, 2023, pp. 208–224.

[5] A. X. Zhang, M. Muller, and D. Wang, “How do Data Science Workers Collaborate? Roles, Workflows, and Tools,” *Proc. ACM Hum.-Comput. Interact.*, vol. 4, no. CSCW1, May 2020.

[6] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, “Using an LLM to Help With Code Understanding,” in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE ’24. New York, NY, USA: Association for Computing Machinery, 2024.

[7] L. Studtmann, S. Aydin, and H. Lichter, “Histree: A Tree-Based Experiment History Tracking Tool for Jupyter Notebooks,” in *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, 2023, pp. 299–308.

[8] A. P. Koenzen, N. A. Ernst, and M.-A. D. Storey, “Code Duplication and Reuse in Jupyter Notebooks,” in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2020, pp. 1–9.

[9] L. Quaranta, F. Calefato, and F. Lanubile, “KGTorrent: A Dataset of Python Jupyter Notebooks from Kaggle,” in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, May 2021.

[10] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, “A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks,” in *2019 IEEE/ACM 16th international conference on mining software repositories (MSR)*. IEEE, 2019, pp. 507–517.

[11] A. P. S. Venkatesh, J. Wang, L. Li, and E. Bodden, “Enhancing Comprehension and Navigation in Jupyter Notebooks With Static Analysis,” in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2023, pp. 391–401.

[12] T. Mondal, S. Barnett, A. Lal, and J. Vedurada, “Cell2Doc: ML Pipeline for Generating Documentation in Computational Notebooks,” in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 384–396.

[13] D. Ramasamy, C. Sarasua, A. Bacchelli, and A. Bernstein, “Visualising Data Science Workflows to Support Third-Party Notebook Comprehension: An Empirical Study,” *Empirical Software Engineering*, vol. 28, no. 3, p. 58, 2023.

[14] M. B. Kery and B. A. Myers, “Interactions for Untangling Messy History in a Computational Notebook,” in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2018, pp. 147–155.

[15] M. B. Kery, B. E. John, P. O’Flaherty, A. Horvath, and B. A. Myers, “Towards Effective Foraging by Data Scientists to Find Past Analysis Choices,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–13.

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

[17] K. Subramanian, J. Maas, and J. Borchers, “Tractus: Understanding and Supporting Source Code Experimentation in Hypothesis-Driven Data Science,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[18] M. Horiuchi, Y. Sasaki, C. Xiao, and M. Onizuka, “Jupysim: Jupyter Notebook Similarity Search System,” in *EDBT*, 2022.