Bug Analysis in Jupyter Notebook Projects: An Empirical Study

Taijara L. Santana[1], Paulo A. da M. Silveira Neto[2], Eduardo S. Almeida[1], Iftekhar Ahmed[3]

[1]Federal University of Bahia, Institute of Computing (IC-UFBA) Salvador – Bahia – Brazil

[2]Federal University Rural of Pernambuco (UFRPE) Recife – Pernambuco – Brazil

[3]University of California, Irvine USA

taijara@gmail.com, paulo.motant@ufrpe.br, esa@rise.com.br, iftekha@uci.edu

Abstract. *Computational Notebooks, such as Jupyter, have been widely adopted in data science for building data-driven code. Despite their popularity, challenges related to software development in these environments still need to be investigated. This study conducts a systematic analysis of bugs and difficulties faced by Jupyter practitioners. A total of 14,740 commits from 105 GitHub projects were mined, and 30,416 Stack Overflow posts were analyzed to identify common issues. Additionally, 19 interviews with data scientists were conducted to gather more details on these challenges. For validation, a survey with various professionals was carried out, along with an analysis based on the Apriori algorithm. Based on these findings, a taxonomy of bugs was proposed to classify different types of issues found in Jupyter projects.*

1. Introduction

Data science and data analysis are emerging fields that attract professionals from various areas, such as mathematics, statistics, and computer science. These professionals combine technical knowledge with domain expertise to obtain strategic insights through data exploration, quantification, qualification, and prediction [Dhar 2013, Tao et al. 2020, Cao 2017]. Jupyter Notebook has become one of the most widely used tools in this context, allowing the writing of text and code in a documentation structure that describes the data analysis process [Pimentel et al. 2019, Koenzen et al. 2020]. However, despite its popularity, Jupyter presents challenges, such as promoting inadequate development practices and reproducibility issues [Wang et al. 2021, Head et al. 2019].

The growing use of Jupyter Notebook has brought benefits but also challenges, such as code incompatible with Python standards, unused variables, and obsolete functions [Wang et al. 2020]. Only 24.11% of the analyzed notebooks could be reproduced without errors [Pimentel et al. 2019], presenting issues like *name-value inconsistency* [Patra and Pradel 2021] and the absence of *dependency* declarations in 94% of cases [Wang et al. 2021]. These factors lead data scientists to perceive notebooks as *ad-hoc* and disposable tools [Kandel et al. 2012], often described as disorganized [Kery et al. 2018, Rule et al. 2018]. Understanding these issues can help improve the reliability and usability of Jupyter notebooks.

Despite extensive research on software bugs in various domains [Thung et al. 2012, Zhang et al. 2018, Islam et al. 2019], studies focusing specifically on Jupyter projects remain scarce. Analyzing bugs in Jupyter notebooks is crucial for improving code quality and reliability [Wang et al. 2021, Chattopadhyay et al. 2020], as these issues can significantly impact data science projects. A notable example is the COVID-19 case, where errors in Jupyter notebooks led to significant discrepancies in data reporting [Insider 2020].

Given these challenges, this work aims to investigate bugs in Jupyter projects, including their characteristics, root causes, impacts, and the challenges faced by data scientists. To achieve this, we analyze commits from GitHub repositories and Stack Overflow posts related to Jupyter bugs [Agrawal and Srikant 1994]. Additionally, interviews with data scientists and a survey were conducted to validate the findings. By shedding light on these problems, we hope to contribute to the development of better tools and practices for Jupyter users, improving both the usability and robustness of computational notebooks.

As a result of this work, the main contributions can be enumerated: Comprehensive understanding of bug classes and their root causes in the context of Jupyter Proposal of a taxonomy with eight bug categories specific to Jupyter projects. Recommendations for researchers and practitioners based on data collected from GitHub, Stack Overflow, and interviews. Availability of materials for reproducible research, including datasets and interview data (available on the project website[1]) .

2. Related Work

The main related work were grouped into five conceptual areas:

Extensions to Improve Notebooks: The Jupyter Notebook project aims to provide the data science community with a simple graphical interface to promote computational narrative, focusing on usability, collaboration, and portability. Several studies have proposed extensions to improve these aspects. For example, [Rule et al. 2018] investigated how ”cell folding”can improve notebook navigation and reading, while [Head et al. 2019] developed a solution to collect and organize code versions. [Kery et al. 2020] created an API for interacting with graphical outputs, and [Wang et al. 2020, Wang et al. 2021] focused on improving notebook reproducibility.

How Data Scientists Use Jupyter Notebooks: Studies also explore how data scientists use notebooks in their daily work. [Koenzen et al. 2020] analyzed code duplication, identifying that although there is an 8% rate of duplicate code in GitHub repositories, users refer not to duplicate their own code. [Wang et al. 2019] studied real-time collaboration, highlighting that synchronous notebooks encourage exploration but require greater team coordination.

Notebook Quality: Notebook quality is a recurring theme. [Chattopadhyay et al. 2020] cataloged nine main problems faced by data scientists when using notebooks. [Rule et al. 2018] analyzed the structure of 1 million notebooks, identifying that most lack proper cleaning and documentation, hindering readability and reproducibility. [Pimentel et al. 2019, Pimentel et al. 2021] conducted large-scale studies on reproducibility issues, showing that only 24.11% of notebooks run without errors.

Empirical Studies on Bugs: Some studies analyze bugs in different domains, such as deep learning [Zhang et al. 2018], IoT [Makhshari and Mesbah 2021], and auto-

[1]`https://github.com/bugsjupyterempiricalstudy/BugJupyterPaper`

nomous vehicles [Garcia et al. 2020]. However, this is the first empirical study focused on bugs in Jupyter Notebook projects.

Studies on Bugs with Taxonomies: Four studies have approaches similar to our research, identifying and characterizing bugs in specific domains, such as autopilot software [Wang et al. 2021] and Infrastructure as Code scripts [Rahman et al. 2020]. These studies used techniques such as data mining, manual classification, and expert validation to build bug taxonomies.

3. Methodology

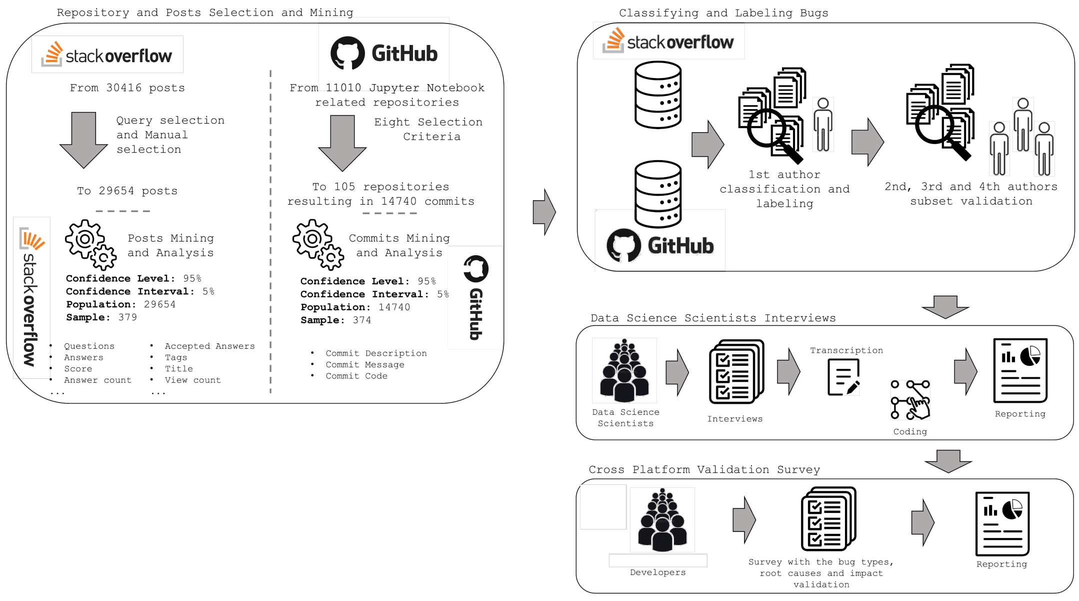
Repository and Posts Selection and Mining
stackoverflow
From 30416 posts
Query selection and Manual selection
To 29654 posts
Posts Mining and Analysis
Confidence Level: 95%
Confidence Interval: 5%
Population: 29654
Sample: 379
Questions
Answers
Score
Answer count
...
Accepted Answers
Tags
Title
View count
...
GitHub
From 11010 Jupyter Notebook related repositories
Eight Selection Criteria
To 105 repositories resulting in 14740 commits
Commits Mining and Analysis
Confidence Level: 95%
Confidence Interval: 5%
Population: 14740
Sample: 374
Commit Description
Commit Message
Commit Code
Classifying and Labeling Bugs
1st author classification and labeling
2nd, 3rd and 4th authors subset validation
Data Science Scientists Interviews
Data Science Scientists
Interviews
Transcription
Coding
Reporting
Cross Platform Validation Survey
Developers
Survey with the bug types, root causes and impact validation
Reporting

Figura 1. Research Methodology.

Research Design and Data Collection. This study seeks to answer four research questions (RQs): *RQ1:* What types of bugs are most frequent? *RQ2:* What are the root causes of bugs? *RQ3:* What are the frequent impacts of bugs? *RQ4:* What challenges do data scientists face when using Jupyter Notebooks? To answer these questions, a total of 105 GitHub repositories written in Jupyter Notebook were selected based on criteria such as the number of stars, contributors, and 14,740 commits, with 855 bug-fixing commits. Additionally, 30,416 posts related to Jupyter Notebooks were collected from StackOverflow, filtered by tags and score. Bugs were manually classified by four experts using two-cycle coding techniques, identifying bug types, root causes, and impacts through the analysis of commit messages, pull requests, and StackOverflow posts. Validation was performed using Cohen's Kappa coefficient, achieving an agreement level of 0.80.

Data Scientist Interviews and Survey. Semi-structured interviews were conducted with 19 data scientists from 12 companies to address challenges in using Jupyter Notebooks. The interviews were transcribed, coded, and resulted in 52 codes, 7 categories, and 5 main challenges. To validate the results and the proposed taxonomy, a survey was conducted with 91 developers, including both open and closed questions. Likert scales were used to assess agreement with bug categories and root causes.

4. Results

The results provide a detailed analysis of the types of bugs found in Jupyter Notebook projects, their root causes, impacts, and the challenges faced by data scientists, as well as an initial taxonomy (Figure 2).

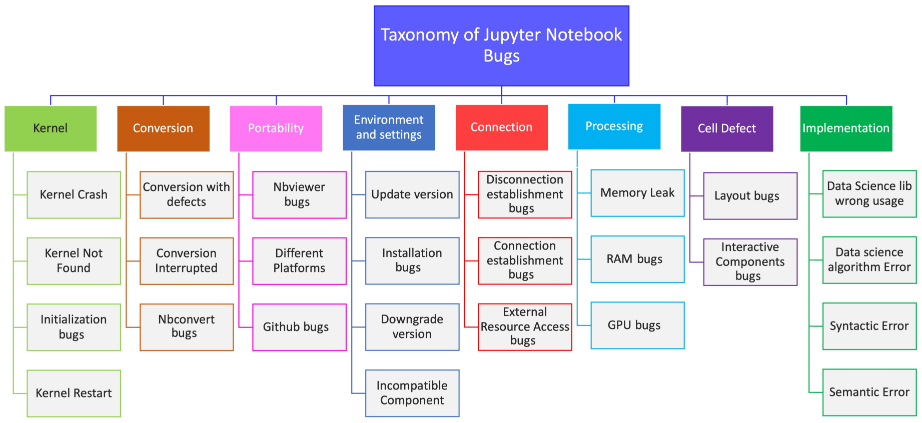

Figura 2. Taxonomy of Jupyter Notebook bugs.

4.1. Types of Bugs in Jupyter Projects (RQ1)

Eight main types of bugs were identified in Jupyter Notebook projects, organized into an initial taxonomy:

Kernel Bugs (KN) – ***(StackOverflow: 10.8% — GitHub: 2.9%)*** This type of bug occurs in kernel operations when using Jupyter Notebooks, including crashes, initialization issues, and unresponsiveness. *Kernel Crash*: The kernel unexpectedly stops working, sometimes displaying an error message. Usually resolved by restarting. *Kernel Not Found*: Happens when Jupyter Notebook fails to link to a kernel, often due to installation issues. *Initialization Bugs*: Errors during kernel startup, usually caused by incorrect installations or conflicts. *Kernel Restart*: The kernel restarts unexpectedly during use. *Example:* These bugs can lead to data loss and project delays. In bug #107937815 (GitHub) and #35673530 (StackOverflow), a Python update caused package incompatibilities, as reported by DS13: ✓ *DS13: "The Kernel bugs are the most frequent ones (...). It impacts project execution time since it interrupts the data analysis."*

Conversion Bugs (CV) - ***(StackOverflow - 6.7% — GitHub - 10.6%)*** These bugs occur during the conversion of `.ipynb` notebooks to other formats, leading to corrupted files or poor rendering. *Conversion Interrupted:* The process stops before completion. *Conversion with Defects:* The file is converted but has issues (e.g., missing images in PDFs). *Nbconvert Bugs:* Errors in the `nbconvert` module prevent conversion from starting. *Example:* Bugs #99244384 (GitHub) and #46415269 (StackOverflow) relate to `nbconvert` issues, impacting user experience, especially for beginners: ✓ *DS12: "It happens with new users, who spend considerable time performing the export procedure."*

Portability Bugs (PB) - ***(StackOverflow - 2.7% — GitHub - 1.3%)*** These bugs occur when running Jupyter Notebooks in different environments, causing compatibility, rendering, or configuration issues. *GitHub Bugs:* Issues rendering `.ipynb` files on GitHub, leading to display errors. *Nbviewer Bugs:* Similar to GitHub, but affecting the nbviewer platform. *Different Platforms:* Errors when running a notebook on a different OS, machine, browser, or platform (e.g., Google Colab, JupyterLab) due to configuration differences. <u>*Example:*</u> These issues hinder analysis sharing. Bugs #200722670 (GitHub) and #47868625 (StackOverflow) highlight GitHub rendering problems. ✓ *DS11: "GitHub has a tool to view Jupyter notebooks, but it's random—it opens whenever it wants. It doesn't always work in the browser."*

Environments and Settings (ES) - *(StackOverflow - 43.2% — GitHub - 35.6%)* These bugs arise from development environment and configuration issues, including missing/deprecated libraries, installation errors, OS incompatibilities, and package manager problems (e.g., Anaconda, PIP). *Update/Downgrade Issues:* Incompatibilities requiring version changes in libraries or extensions. *Installation Bugs:* Errors due to incorrect installation or missing dependencies. *Incompatible Components:* Conflicts between notebook components or extensions. <u>*Example:*</u> Setting up the environment is time-consuming. Bugs #200722670 (GitHub) and #35561126 (StackOverflow) show issues caused by Python version mismatches. ✓ *DS13: "Depending on the project and dependencies, environment setup is laborious. Jupyter Notebooks could help manage it, avoiding configuration problems."*

Connection Bugs — CN - *(StackOverflow - 6.2% — GitHub - 0.9%)* These bugs occur when connecting notebooks to external resources like databases, hardware, and repositories. *External Resource Access Bugs*: Notebook loses access to external resources. *Disconnection Bugs*: Notebook loses connection to its server. <u>*Example:*</u> Bugs #107937815 (GitHub) and #63863571 (StackOverflow) involve URL and external image connection issues. DS2 highlighted similar problems when using Arduino: ✓ *DS2: "... using Arduino, we tried several workarounds—disconnecting, reconnecting, replacing the board—until it finally connected to Jupyter."*

Processing — PC - *(StackOverflow - 4.9% — GitHub - 1.9%)* Processing bugs involve memory issues, timeouts, and long-running tasks. *Memory Leak*: Inefficient memory allocation causing execution delays. *RAM and GPU Bugs*: Memory overflow and slow processing. <u>*Example:*</u> These bugs increase analysis time and cause data loss. Bugs #86884600 (GitHub) and #643288550 (StackOverflow) report high-resolution image processing issues. Chattopadhyay et al. [Chattopadhyay et al. 2020] also noted Jupyter's limitations in handling large datasets, echoed by DS10: ✓ *DS10: "It happened several times while handling large datasets. Debugging and identifying the root cause took time."*

Cell Defect (CD) - *(StackOverflow - 3.6% — GitHub - 2.6%)* Bugs related to notebook cell rendering, including code, markdown, and outputs, often occurring with interactive components, LaTeX, and graphics. *Layout Bugs*: Issues like text overflowing, unexpected formulas, blank cells, and visualization errors. *Interactive Components Bugs*: Errors with interactive elements within cells. <u>*Example:*</u> Bugs #237890763 (GitHub) and #69695030 (StackOverflow) report issues with "input()"and scrollbars. DS14 mentioned: ✓ *DS14: "For some reason, the cell size reduced and ended up cutting the text in half. I couldn't identify the cause, and it happens a lot."*

Implementation — IP - *(StackOverflow - 22% — GitHub - 44.2%)* Bugs related to syntax, logic, uninstantiated variables, algorithms, and semantics. *Semantic Error*: Code executes but produces incorrect output due to logic mistakes. *Syntax Error*: Issues like incorrect variable/function declarations, missing/misplaced parentheses, and nonstandard Python (PEP8). *Data Science lib wrong usage*: Misuse of functions from libraries like Pandas, Scikit-learn, and TensorFlow. *Data Science Algorithm Error*: Errors in statistical analysis or machine learning logic. *Example:* Bugs #222507066 (GitHub) and #45946060 (StackOverflow) involve duplicate code and out-of-order execution. DS14 noted: ✓ *DS14: "When writing in a notebook, it's easy to lose context. Running cells out of order makes everything confusing."*

4.2. Root Causes of Bugs (RQ2)

Understanding the root causes of bugs helps identify their origin and potential solutions. Table 1 presents the distribution of bug types by root cause. In the following, we describe each category and its percentage of occurrence.

Root Causes	**ES**		**IP**		**KN**		**CV**		**CN**		**PC**		**CD**		**PB**		Total
	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	
Install and Configuration Problems	635	117	36	0	56	12	44	0	42	0	0	0	6	7	12	3	970
Coding error	3	0	20	0	34	0	2	0	7	0	78	12	0	1	1	0	724
Version Problems	17	18	397	185	4	3	12	49	4	4	0	3	17	7	4	0	682
Unknown	16	0	3	0	2	0	1	0	1	1	1	0	0	0	0	0	413
Hardware software limitations	21	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	224
Logic error	4	0	7	0	5	0	9	0	18	0	6	1	0	0	0	0	169
Memory Error	2	1	32	114	9	1	0	0	0	0	4	0	4	2	0	0	158
TimeOut	12	0	15	0	5	0	20	39	19	1	36	0	33	4	32	8	50
Deprecation	374	158	0	25	45	7	24	2	30	0	0	0	11	0	6	0	25
Permission denied	34	9	58	54	117	2	60	1	38	2	1	0	22	1	14	0	25
Total	1118	304	569	378	278	25	172	91	160	8	126	16	93	22	69	11	

(KN) Kernel, (CV) Conversion, (PB) Portability, (ES) Environment and Settings, (CN) Connection, (PC) Processing, (CD) Cell Defect, (IP) Implementation.

Tabela 1. Frequency of Bug Type vs Root Cause

Install and Configuration Problems - ***(StackOverflow - 32.1% — GitHub - 16.3%)***. These bugs arise when setting up the development environment, often leading to process interruptions and productivity loss. Due to the exploratory nature of data analysis, users frequently require additional tools and configurations, increasing the likelihood of these issues.

Version Problems - ***(StackOverflow - 19.0% — GitHub - 22.5%)*** and **Deprecation -** ***(StackOverflow - 0.9% — GitHub - 0.1%)***. Incompatible software versions necessitate updates or downgrades, often resulting in conflicts. Unlike classic IDEs, Jupyter Notebook lacks intelligent version controls, leaving users to manage dependencies manually. Deprecation-related bugs occur when discontinued components or features cause issues.

Coding Errors - ***(StackOverflow - 17.6% — GitHub - 31.5%)*** and **Logic Errors -** ***(StackOverflow - 2.0% — GitHub - 13.7%)***. Coding errors involve incorrect variable assignments, redundant structures, or library misuse, typically manifesting as runtime errors. Logic errors, on the other hand, stem from flawed code logic, impacting results without necessarily causing runtime failures.

Hardware and Software Limitations - *(StackOverflow - 6.7% — GitHub - 6.1%)* and **Memory Errors - *(StackOverflow - 5.6% — GitHub - 1.5%)***. Performance constraints, such as slow execution or failures, often arise from hardware or software limitations. Memory errors, caused by overflows, lead to crashes and require re-execution.

Permission Denied - *(StackOverflow - 0.9% — GitHub - 0.1%)* and **TimeOut - *(StackOverflow - 1.9% — GitHub - 0.1%)***. Permission errors occur when Jupyter Notebook is blocked from accessing external resources, such as databases. Timeout issues arise when processes take too long to execute, resulting in crashes.

Unknown - *(StackOverflow - 13.3% — GitHub - 8.1%)*. These bugs lack clear root causes and often have no effective solutions.

Tables 1 highlights that the most frequent root causes are Install and Configuration Problems (StackOverflow - 32.1% — GitHub - 16.3%), Version Problems (StackOverflow - 19.0% — GitHub - 22.5%), and Coding Errors (StackOverflow - 17.6% — GitHub - 31.5%). These issues mainly stem from configuration errors, incompatible versions, and coding mistakes, significantly affecting productivity and workflow.

4.3. Impacts of Bugs (RQ3)

The impact caused by a bug can help increase its severity and serve as a prioritization model and alert for users. Tables 2 shows the distribution of bug types according to their impact. Below, we describe each category and its occurrence percentage.

Impacts	**ES**		**IP**		**KN**		**CV**		**CN**		**PC**		**CD**		**PB**		**Total**
	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	SO	GH	
Run Time Error	900	197	1	0	275	11	49	0	142	0	8	1	4	0	8	0	1753
Incorrect Functionality	41	90	18	46	1	0	2	0	2	0	46	14	1	3	0	0	840
Crash	142	16	64	261	0	10	96	91	7	8	1	0	86	19	55	11	657
Bad Performance	7	1	472	65	2	4	24	0	9	0	71	1	2	0	6	0	141
Warning	28	0	14	6	0	0	1	0	0	0	0	0	0	0	0	0	49
Total	1118	304	569	378	278	25	172	91	160	8	126	16	93	22	69	11	

(KN) Kernel, (CV) Conversion, (PB) Portability, (ES) Environment and Settings, (CN) Connection, (PC) Processing, (CD) Cell Defect, (IP) Implementation

Tabela 2. Frequency of Impact vs Root Cause

Run Time Error - *(StackOverflow - 57.5% — GitHub - 31.2%)* and **Incorrect Functionality - *(StackOverflow - 13.5% — GitHub - 57.3%)***. Run Time Errors are characterized by execution failures, often accompanied by error messages, and are the most frequent impact in the StackOverflow dataset. Incorrect Functionality bugs produce unwanted or incorrect outputs, being the most common impact on GitHub.

Crash - *(StackOverflow - 24.3% — GitHub - 3.3%)*. These bugs cause a break in one or more components of the platform, interrupting its operation or initialization. Kernel Crash, a common issue, often requires restarting the kernel to resolve.

Bad Performance - *(StackOverflow - 3.0% — GitHub - 7.5%)* and **Warning - *(StackOverflow - 1.7% — GitHub - 0.7%)***. Bad Performance bugs do not prevent execution but degrade quality or performance, while Warning bugs trigger alerts without affecting functionality.

The most frequent impacts from bugs in Jupyter notebooks are: Run Time Error (StackOverflow - 57.5% — GitHub - 31.2%), Incorrect Functionality (StackOverflow

- 13.5% — GitHub - 57.3%), and Crash (StackOverflow - 24.3% — GitHub - 3.3%). These impacts are primarily associated with bug types such as Environments and Settings, Implementation, and Kernel. Kernel Crash, a common issue, is often resolved by restarting the kernel, as highlighted by users.

4.4. Challenges in Jupyter Notebook Projects (RQ4)

Data science is a multidisciplinary field, and Jupyter Notebook users come from diverse backgrounds, including physics, mathematics, statistics, and IT. This diversity influences how users perceive and handle bugs, leading to several challenges. Below, we discuss the main issues identified by professionals during interviews.

Backgrounds, Code Quality, and Programming Practices. The diversity of user backgrounds significantly impacts how bugs are identified and fixed. Users without a computing background often produce messy notebooks, leading to errors. The simplistic layout of Jupyter, compared to traditional IDEs, can discourage the use of development standards that improve code quality. As one professional noted: *✓DS3: "Data science combines statistics and computing. People from physics or engineering may not prioritize structured, readable, or documented code, unlike those with a software engineering background."*

This lack of emphasis on code quality is exacerbated by Jupyter's flexibility, which allows users to duplicate and rearrange cells without enforcing good software engineering practices. Many users, especially those new to programming, develop bad habits, such as neglecting testing and linting. As highlighted by: *✓DS14: "I write better code in IDEs like VSCode or RStudio than in Jupyter. The lack of functionality discourages good practices."*

Testing, Debugging, and Bug Resolution. The absence of basic testing and debugging tools in Jupyter is a significant challenge. Users often rely on trial and error to fix bugs, and those with a software engineering background miss features like linting and unit testing. This lack of support makes it harder to identify and resolve issues, as reflected in StackOverflow data, where only 27.9% of questions receive accepted answers, and the average time to resolve a bug is 21 days. *✓DS11: "I miss unit testing and linting in Jupyter. Tools like Black, Pylint, and Flake8 are hard to use in notebooks, which affects code quality."*

Deployment and Production Challenges. While Jupyter is excellent for exploratory analysis and prototyping, it lacks tools to facilitate deployment into production systems. Many interviewees reported challenges in converting notebook code into deployable products. This gap is particularly problematic for industrial projects, where Jupyter notebooks are often integrated into larger systems. *✓DS7: "Deploying Jupyter code into a system is not trivial. Tools to support this process would be valuable."*

Data scientists perceive bugs differently based on their experience with software engineering, and the lack of testing and debugging tools in Jupyter makes it harder to identify and fix bugs, leading to longer resolution times and lower acceptance rates on platforms like StackOverflow. Additionally, transforming Jupyter analyses into deployable products is challenging due to the absence of tools to improve code quality and support deployment, which can lead to bad programming habits, especially among beginners.

Enhancing Jupyter with features that promote good practices and facilitate deployment is crucial for its use in industrial and production environments.

5. Conclusion

In this work, we proposed a taxonomy of Jupyter notebook-specific bugs by analyzing these bugs. In particular, we identify eight classes of bugs, ten types of root causes, and the impact of bugs. The most frequent bugs in the Jupyter notebook are those related to Environments and Settings and Implementation. Regarding the root causes, the most frequent were: Configuration issues, Version issues, and Coding Errors. They are the cause of most Implementation and Environments and Settings bugs. The most frequent bug impact was Run Time Error, followed by Incorrect Functionality. In addition, we found that the data scientist's background determines how the bugs are identified, highlighting the importance of testing and debugging tools. Finally, we identified the Jupyter notebook deployment as a challenging and poorly supported task.

We believe this study can facilitate practitioners' understanding of the nature of bugs and define possible strategies to mitigate them. Our findings can guide future research in related areas, such as developing tools for detecting and recommending bug fixes in the Jupyter notebook and an empirical study to understand the issues in private projects.

References

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *International Conference on Very Large Data Bases*, pages 487–499.

Cao, L. (2017). Data science: A comprehensive overview. *ACM Comput. Surv.*

Chattopadhyay, S., Prasad, I., Henley, A. Z., Sarma, A., and Barik, T. (2020). What's wrong with computational notebooks? pain points, needs, and design opportunities. In *CHI '20*, pages 1–12.

Dhar, V. (2013). Data science and prediction. *Commun. ACM.*

Garcia, J., Feng, Y., Shen, J., Almanee, S., Xia, Y., and Chen, Q. A. (2020). A comprehensive study of autonomous vehicle bugs. In *ICSE '20*, pages 385–396.

Head, A., Hohman, F., Barik, T., Drucker, S. M., and DeLine, R. (2019). Managing messes in computational notebooks. In *CHI Conference*, page 270.

Insider, B. (2020). Thousands of coronavirus cases were not reported for days in the uk because officials exceeded the data limit on their excel spreadsheet.

Islam, M. J., Nguyen, G., Pan, R., and Rajan, H. (2019). A comprehensive study on deep learning bug characteristics. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 510–520.

Kandel, S., Paepcke, A., Hellerstein, J. M., and Heer, J. (2012). Enterprise data analysis and visualization: An interview study. *IEEE Trans. Vis. Comput. Graph.*, 18:2917–2926.

Kery, M. B., Radensky, M., Arya, M., John, B. E., and Myers, B. A. (2018). The story in the notebook: Exploratory data science using a literate programming tool. In *CHI Conference*, page 174.

Kery, M. B., Ren, D., Hohman, F., Moritz, D., Wongsuphasawat, K., and Patel, K. (2020). mage: Fluid moves between code and graphical work in computational notebooks. In *UIST '20*, pages 140–151.

Koenzen, A. P., Ernst, N. A., and Storey, M. D. (2020). Code duplication and reuse in jupyter notebooks. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 1–9.

Makhshari, A. and Mesbah, A. (2021). Iot bugs and development challenges. In *International Conference on Software Engineering*, pages 460–472.

Patra, J. and Pradel, M. (2021). Nalin: Learning from runtime behavior to find name-value inconsistencies in jupyter notebooks. In *International Conference on Software Engineering*.

Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2019). A large-scale study about quality and reproducibility of jupyter notebooks. In *International Conference on Mining Software Repositories*, pages 507–517.

Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2021). Understanding and improving the quality and reproducibility of jupyter notebooks. *Empir. Softw. Eng.*

Rahman, A., Farhana, E., Parnin, C., and Williams, L. (2020). Gang of eight: A defect taxonomy for infrastructure as code scripts. In *International Conference on Software Engineering*, pages 752–764.

Rule, A., Tabard, A., and Hollan, J. D. (2018). Exploration and explanation in computational notebooks. In *CHI Conference*, page 32.

Tao, Y., Jiang, J., Liu, Y., Xu, Z., and Qin, S. (2020). *Understanding Performance Concerns in the API Documentation of Data Science Libraries*, pages 895–906.

Thung, F., Wang, S., Lo, D., and Jiang, L. (2012). An empirical study of bugs in machine learning systems. In *International Symposium on Software Reliability Engineering*, pages 271–280.

Wang, A. Y., Mittal, A., Brooks, C., and Oney, S. (2019). How data scientists use computational notebooks for real-time collaboration. *Proc. ACM Hum.-Comput. Interact.*

Wang, D., Li, S., Xiao, G., Liu, Y., and Sui, Y. (2021). An exploratory study of autopilot software bugs in unmanned aerial vehicles. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 20–31.

Wang, J., Li, L., and Zeller, A. (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*, pages 53–56.

Zhang, Y., Chen, Y., Cheung, S.-C., Xiong, Y., and Zhang, L. (2018). An empirical study on tensorflow program bugs. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 129–140.