This preprint is a preliminary version. The final version will be published at APSEC 2023.

HISTREE: A Tree-Based Experiment History Tracking Tool for Jupyter Notebooks

Laurens Studtmann
Research Group Software Construction
RWTH Aachen University
Aachen, Germany
laurens.studtmann@rwth-aachen.de

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*—When experimenting on solutions to Machine Learning problems, data scientists often integrate non-linear workflows into Jupyter Notebooks to explore different approaches and evaluate the impact of changes such as using different Machine Learning models or adjusting parameters. This mode of working leads to Notebooks that are cluttered and difficult to navigate which complicates refining and reusing previous experiments later on.**

Jupyter Notebooks lack inherent support for such a mode of working. Therefore, we propose the JupyterLab extension HISTREE that provides an interactive tree-based representation of the experiment history in Jupyter Notebooks. Hereby, Notebook versions triggered by specified Notebook operations, are automatically saved and arranged in a tree structure. In this way, HISTREE allows data scientists to explore, compare, organize, and refine their past experiment approaches.

In this paper, first, we introduce the concept of an experiment history tree model. This is followed by a comprehensive description of the functionality of HISTREE, which aims to support data scientists in organizing experiments in Jupyter Notebooks. Initial feedback from user experiments shows that the tree-based experiment model is very promising and that the HISTREE extension is both useful and usable to conduct ML experiments in Jupyter Notebooks.

***Index Terms*—Data Science, Exploratory Programming, Jupyter Notebook**

I. INTRODUCTION

Trial and error is a fundamental problem-solving method in science, often used by data scientists when solving a Machine Learning (ML) problem. Hereby, they iteratively experiment with different solution approaches, techniques, or parameters. They try various methods, observe the outcoming model performance metrics, and – based on the results – adjust their approach until they reach a satisfactory solution. This experimentation process can therefore be modeled by a tree structure, caused by frequent backtracking [1] [2].

Among the most popular tools for exploring ML problem solutions are Jupyter Notebooks [3] [4]. A Jupyter Notebook provides an interactive and flexible development environment in a document-like structure, in which data scientists can add text, executable code snippets, and other kinds of media in the form of cells. The outputs of the code cells are presented below the corresponding cell. This way, data scientists can document what and for which reasons something is done in their experiments.

However, the flexible, iterative nature of exploratory programming often results in cluttered Notebooks that do not adhere to best practices [5] [6].

A study conducted by Ramasamy et al. [7] analyzing 470 real-world Jupyter Notebooks showed that 81.1% of the Notebooks did not follow a linear experiment workflow and 30% of the non-linear experiment workflows had at least three potential divergence points.

There are several reasons for including multiple experiment workflows in a single Notebook. Data scientists may need to explore different solution approaches and compare the effects of changes in an experiment, such as using different kinds of ML models or parameters [1] [2] [8]. Additionally, they may want to revisit a previous experiment for further refinement or reuse.

These are some examples of user requirements that Jupyter Notebooks do not natively support. Based on existing studies, we have identified the following major user requirements for experimentation tools like Jupyter Notebooks that data scientists use when conducting non-linear experimentation workflows. Such tools should provide abilities to:

R1: visually organize different experiments [8] [7] [9]
R2: keep track of an experiment, the written code, and execution results [1] [8] [9] [10]
R3: explore past experiments [1] [8] [11], i.e.
  a. comprehend the experiment workflow history,
  b. compare the results of different experiments,
  c. select a previously performed experiment and further refine it.

In addition, the following usability requirements should be met by such tools as well:

R4: Tracking experiments should require minimal effort by data scientists.
R5: The amount of information tracked and its visualization should not overwhelm data scientists [1].

With regard to the implementation of these requirements, this paper makes the following contributions:

- A conceptual tree-based experiment model based on Jupyter Notebook versions.

- A JupyterLab extension, called HISTREE, for experiment history tracking, implementing the tree-based experiment model and a set of features supporting experimentation.
- The results of user experiments on HISTREE to give insights into usability and acceptance among Jupyter Notebook users.

The paper is structured as follows. First, we discuss related work in Section II. After that, we introduce the tree-based experiment history model in Section III before we dive deeper into its application in Jupyter Notebooks in Section IV. Following this, we describe HISTREE and its features in Section V. Next, we present the results of user experiments in which participants were asked to perform experiments using HISTREE. Finally, the paper concludes with a summary of our contributions and some remarks regarding future research.

II. RELATED WORK

In this section, we give an overview of tools that data scientists use for experimentation. We introduce and analyze the limitations of the tools VERDANT, VARIOLITE, TRACTUS, and NEXTJOURNAL that provide partial history tracking for ML experiments. Additionally, we briefly present the tools TRACTUS and MARG that utilize a tree-based structure to visualize information in ML experiments.

VERDANT*:* This JupyterLab extension designed to help with foraging for information in Notebooks was developed by Kery et al. [12] [13]. VERDANT captures and stores the history of cells and their outputs while working in a Notebook, and saves them as a separate file.

Each change will be tracked using a checkpoint. Checkpoints are assigned version numbers, which are further grouped by days. Each checkpoint is presented visually through a representation known as a mini-map. The mini-map displays a small vertical line for every modified cell within the checkpoint, with each line's color representing the type of change made. The horizontal positioning of the line in the mini-map corresponds to the location of the modified cell in the Notebook. By hovering the cursor over a line, data scientists can retrieve the precise cell number that changed. Furthermore, clicking on the version number opens a read-only snapshot (ghost book) of the Notebook at that time in a new tab and visualizes the changes made. Ghost books have to be exported into the Jupyter Notebook format to be modified.

To make it easier for data scientists to search the history of a Notebook, VERDANT allows data scientists to search for any text within cells and outputs that are or were once part of the Notebook. Furthermore, VERDANT provides the ability to inspect the history of individual cells or outputs, enabling data scientists to quickly locate specific information without going through the entire Notebook history.

The developers of VERDANT conducted a study where participants were asked to perform code foraging tasks in a Jupyter Notebook using VERDANT. The study identified the difficulty of navigating between the different features of VERDANT, which are spread across multiple menus. Although revised versions of VERDANT have attempted to simplify the navigation, the authors suggest that a more comprehensive investigation into the interface design would be required to address this issue.

However, one area where VERDANT does not provide direct support is experimenting with alternative versions of code and branching to keep track of those versions (R1). There is no proper support for continuing work on previous versions of the Notebook (R3c). While it is possible to export ghost books, this generates a separate Notebook, making it impractical for addressing quick and immediate local versioning needs.

VARIOLITE*:* To better assist data scientists in their need for versioning, Kery et al. developed VARIOLITE, an Atom code editor extension [11]. VARIOLITE introduces variant boxes that can be placed around code snippets. These boxes enable the creation of alternate versions of the code, switchable through attached tabs. Although VARIOLITE offers some basic experiment tracking, its primary focus is on versioning code snippets rather than comprehensive file and project history. The Atom code editor is deprecated since 2022.

NEXTJOURNAL*:* This is a cloud-based platform designed for executing computational Notebooks that are compatible with Jupyter Notebooks [14]. One distinguishing feature that sets it apart from other platforms is its comprehensive history tracking capability. This includes not only tracking the content of the Notebook itself but also monitoring the entire computational environment and associated data. However, it should be noted that while data scientists can navigate between different versions, the history tracking system is linear in nature, similar to VERDANT. It does not integrate history as a fundamental element for organizing experiments but rather offers basic functionality for undoing changes.

TRACTUS*:* The main concept behind TRACTUS, an RStudio addin created by Subramanian et al. [10], is to view code for data science applications as a series of experiments. TRACTUS operates within the programming language R, specifically designed for data science, and organizes the data scientist's experiments into hypotheses. These hypotheses are statements that the data scientist aims to verify as either true or false. The addin arranges these hypotheses in a tree structure and provides visualizations to help the data scientist better recall and understand the events that occurred in the code and the significant insights gained from the results.

MARG*:* In contrast to the previous tools, the JupyterLab extension MARG [15] is not providing any history tracking capabilities but offers means for the narrative exploration of non-linear workflows in a Notebook. Hereby, workflows are visualized as ordered trees. Each node in the tree corresponds to a Notebook cell and shows additional information, e.g. what kind of activity is performed in the cell and the execution number. Each path from the root to the leaf corresponds to one linear workflow. Divergences from linear workflows are depicted by branches in the tree. By clicking through nodes in the tree, data scientists can better understand and

navigate the structure of the Notebook and how cells are forming workflows. In their paper, the authors use a static representation of the tree for a certain example, a dynamic generation of the tree while working in a Notebook is in planning.

In general, all existing history tracking tools support only a linear history or are not applicable to Jupyter Notebooks. This highlights the need to support data scientists' exploratory and non-linear approach to experimentation with additional tools.

Furthermore, there have been positive outcomes in terms of usability and acceptance in user studies of tools like TRACTUS and MARG which employ a tree-based representation of experiments.

This motivates us to develop a tree-based concept and implementation for handling and visualizing the experiment history in Jupyter Notebooks which we will present in the following.

III. The Tree-based Experiment History Model

To solve an ML problem, data scientists usually conduct a series of experiments. The experiments are evaluated and improved until a solution is found that meets the requirements. The following typical example describes a simplified workflow when searching for a solution to a clustering problem.

The data scientist starts the experiment by loading the data, then selecting a few features. Next, they try out k-means, a popular clustering algorithm. Finally, an evaluation is performed. As the evaluation of the resulting clusters is not satisfactory, the data scientist decides to try out DBScan, a different clustering algorithm. Therefore, the data scientist goes back to the start of the experiment, loads the same data, and selects the same features as they did in the first experiment. However, now the DBScan algorithm is applied. As the evaluation results of this new experiment are better but still leave room for improvement, the data scientist decides to keep DBScan but change the selected features. To do so, the data scientist goes back again to the start of the experiment, loads the same data but now selects a different set of features before using the DBScan algorithm again. As the obtained results are now very good, a solution is found.

This way of working can be mapped onto the following tree-based model for experiments:

- An *experiment step* is an atomic experiment action or a semantic collection of related actions to achieve a particular objective; experiment steps are ordered.
- An experiment step has no or exactly one predecessor. If a step has no predecessor, it is called *start step*.
- An experiment step may have successors. If a step has no successor, it is called a *finish step*. It is called a *branching step* if it has more than one successor.
- An experiment consists of a linear sequence of experiment steps (e.g. loading data, evaluating results). The first step is a start step, and the last one is a finish step.

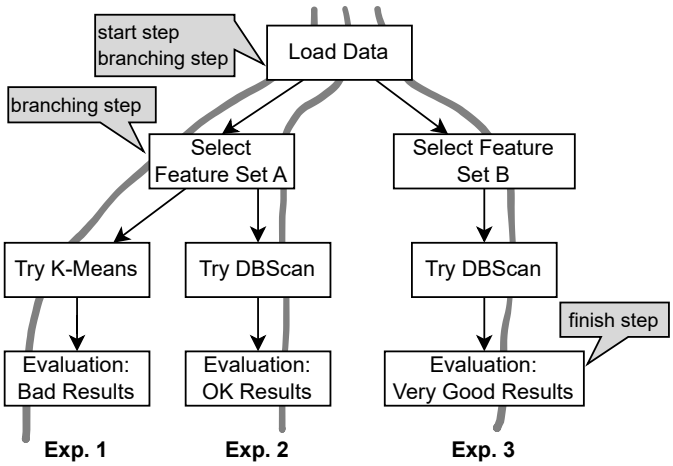

Fig. 1: Example of an experiment history tree

- All experiments that have a common branching step are called *variants* related to this branching step. This means that variant experiments have common experiment steps (the ones from the start step to the branching step) and different experiment steps (the ones after the branching step to the experiment's finish steps).

If we transform the described workflow into an instance of this model, we get the *experiment history tree* shown in Figure 1. As we can see, the experiment history tree models three experiments. All experiments are variants related to the step `Load Data`, which is the start step of all experiments and also a branching step. Furthermore, experiments 1 and 2 are variants related to the `Select Feature Set A` step. The evaluation steps are the finish steps of the experiments.

IV. The Experiment History Tree in Jupyter Notebooks

When data scientists use Jupyter Notebooks to conduct experiments, all kinds of Notebook *modification operations* are performed, e.g. cells are added or a new statement is typed into a code cell. Each modification operation results in a new version of the Notebook. A *Notebook version* is a timestamped copy of the same; all versions of a Notebook are time-ordered.

To capture the experiments carried out in a Notebook in an experiment history tree, it is not necessary to consider all possible Notebook versions. It is sufficient to consider only those modification operations and the corresponding Notebook versions that are relevant in the sense of conducting the experiments.

As the implementation of an experiment step can be mapped onto one Notebook cell or a sequence of cells, the following modification operations are relevant for creating the experiment history tree that captures all conducted experiments of a Notebook:

- $insertCell$: inserts a new cell to implement an experiment step or a part of it

- $runCell$: executes the code of a cell, which corresponds to the execution of an experiment step
- $runCells$: executes a sequence of cells, which corresponds to the execution of an experiment step consisting of several parts
- $deleteCell$: removes a cell, which corresponds to removing an experiment step
- $moveCell$: moves a cell further up or down, which changes the sequence of an experiment step or a part of it
- $changeCellType$: changes the type of a cell (code, text, raw). Changing the type also changes the experiment step as e.g. code may be transformed to text/raw output or vice versa.

Besides these modification operations, the Notebook creation operation new is relevant. It opens a new Notebook containing an empty cell, which usually implements the start step.

Formally, an experiment history tree is an ordered, labeled tree consisting of nodes; each node represents exactly one Notebook version. The nodes are labeled by a timestamp together with the modification operation identifier that results in the Notebook version represented by the node.

A. Automatic Experiment History Tree Creation

Opening a new Notebook, i.e. performing the new operation, creates the *root node* of the experiment history tree, which represents the first version of this Notebook. The node representing the currently editable Notebook version is marked by a pointer, called the *current node* of the experiment history tree.

The creation process of the experiment history tree follows the principle that each performed modification operation which leads to a new Notebook version is represented in the experiment history tree by adding a child node to the current node. The label of the new child node indicates the type of operation that has been performed. For example, if a cell has been executed, then a new node representing a new Notebook version is added. As this was created by a *runCell* operation, the new node is marked with runCell. If, for example, a cell has been executed, a new node marked by a runCell operation is added as a child node to the current node. The current node is automatically set to the last recently added node.

When the current node is set to a different node of the experiment history tree by the Notebook user, the Notebook version that is represented by this node is loaded and can be edited. If a new modification operation is performed in this Notebook version, the resulting new Notebook version is represented by a new child node added to the existing child nodes of the current node. This always creates a new experiment variant.

This process can be described as follows:

- When a new Notebook is opened, the root node of the experiment history tree is created and the current node points to the root node.

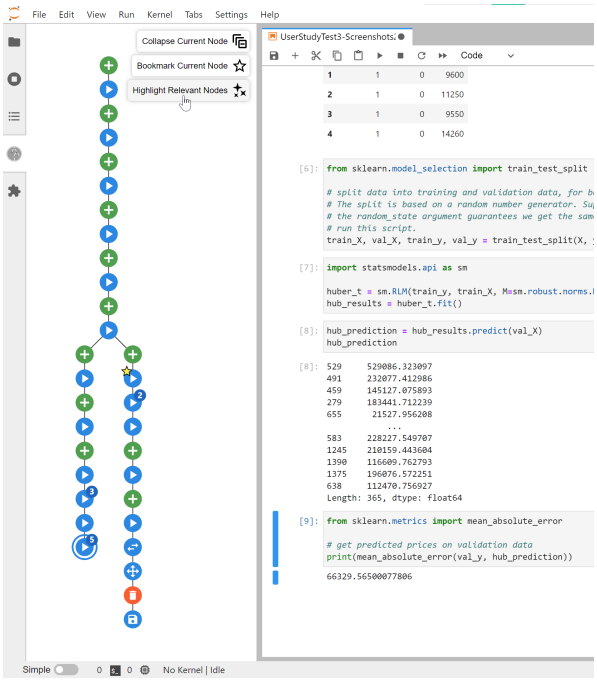

Fig. 2: JupyterLab with installed HISTREE extension

- When a modification operation is performed in the currently editable Notebook version (represented by the current node of the tree) the current node receives a new child node. If the current node already has child nodes, then the new child node is added to the right of the existing child nodes.

Because of the timestamps, the experiment history tree is temporally ordered. Along each path from the root node to the leaf nodes, each child node has a more recent creation timestamp than its parent. Along each child level of a parent node, each child node has an older timestamp than its sibling nodes to the right.

V. HISTREE - A JUPYTERLAB EXTENSION

To make the experiment history model and tree available for Jupyter Notebook users, we have developed the JupyterLab extension HISTREE[1]. HISTREE offers serveral new features which aim to meet the aforementioned user requirements (R1-R3) while contributing to the usability requirements R4 and R5. A screenshot of JupyterLab with the HISTREE extension open in the sidebar can be seen in Figure 2.

The sidebar displays the experiment history tree which can be enlarged and reduced by scrolling and moved by dragging with the mouse, similar to a digital map. The tree visualizes the history of the Notebook, each node representing an experiment step, which is stored in its own Notebook version. Unique icons, possibly decorated, are used to quickly see which modification operation was performed in the respective experiment step (see Table I).

[1]HISTREE makes use of a significantly modified version of the VERDANT JupyterLab extension [13] as the backbone for history tracking.

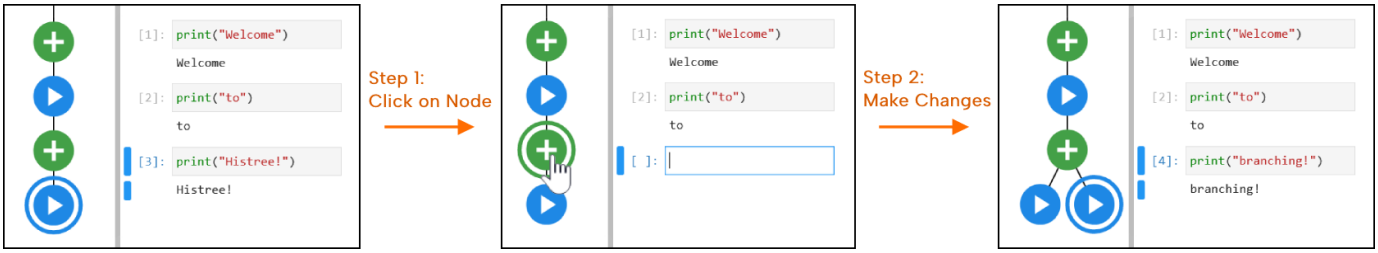

Fig. 3: Steps to create an experiment branch

TABLE I: Node icons and decorations. The decorations can be applied to any node type and may also occur in combination.

Icon	Version whose last modification operation is:
	a new cell was inserted (*insertCell*)
	a cell was executed (*runCell*)
8	a sequence of cells was executed (*runCells*)
	a cell was deleted (*deleteCell*)
	a cell was moved (*moveCell*)
	the type of a cell was changed (*changeCellType*)
Decoration	
	current node
	bookmarked node
	highlighted node

A. Offered Features

In the following, we briefly present the features of HISTREE, subsequently referenced by F1 to F5.

Switching Notebook Versions (F1): HISTREE enables the data scientist to change to a different Notebook version by clicking on a node in the tree. As a consequence, the Notebook will be changed to reflect the state of the Notebook as it was in the version that corresponds to the clicked node. In addition, the clicked node now becomes the current node, indicated by a circle around the node.

Switching to a different Notebook version is a necessary feature to explore past experiments (R3), e.g. to look at previous experiment results to compare (R3b) or reuse them (R3c). Going through the different versions in the history of the Notebook also helps to comprehend how exactly the results of the Notebook were generated by the numerous executions that happened in its history (R3a).

Branching (F2): By switching to and then modifying a previous Notebook version, a branch in the experiment history tree is created. This allows the data scientist to work on an alternative experiment approach without having to overwrite any previous cells or cluttering the Notebook with too many different approaches. As all other variants of the experiment are then available as different branches, the data scientist can switch between them for performing comparisons or continuing work on the most promising approach. The procedure to create a branch is shown in Figure 3. First (Step 1), the data scientist clicks on the node they wish to branch out of, causing that node to become the current node. Then (Step 2), they perform a modification in the Notebook which causes a new node to appear as the first node of the new experiment branch. In the shown example, a print statement was added to an empty cell and then executed. This caused a new node created by the *runCell* operation to be added to the experiment history tree in a new branch.

Modification Summary (F3): Whenever the data scientist moves the mouse cursor over a node in the tree, a modification summary appears. A simple example is shown in Figure 4. The modification summary displays the type of modification operation – like *Cell 3 was executed* – that lead to a new version, the time and date when the modification operation was performed, and a diff-style summary of the changes that happened within the modified cell and its output. If multiple changes have occurred within a version, as is the case with a *runCells*-operation, all changes are listed in the modification summary.

If the data scientist wants to move their cursor over the displayed modification summary, e.g. to copy parts of it, the summary can be pinned by clicking on a small pin needle icon that appears next to the node the cursor is currently hovering over.

Fig. 4: The modification summary feature in use

Bookmarking (F4): To make it easier to find important nodes in the tree, nodes can be bookmarked using the context menu by right-clicking the node or with the bookmark-star button in the top right corner of the sidebar, which will bookmark the current node. Bookmarking a node will add a small yellow star decoration in its top left corner to signify its importance.

Relevant Node Highlighting (F5): This feature can be activated by pressing the *Highlight Relevant Nodes*-button with a spark icon in the top right corner of the sidebar. While activated, those nodes that have had some sort of effect on the currently selected cell will be highlighted using an orange glow around each node. An example is shown in Figure 5. In the currently selected cell, the variable `features` has been assigned. All nodes that have contributed to the creation, deletion or modification of this cell are highlighted with an orange glow.

The purpose of this feature is to help with finding particular nodes in the tree, as the data scientist often wants to look at or branch out of a previous Notebook version where they know that a particular cell has been added or modified. With the relevant node highlighting activated, they can simply select the cell by clicking on it which will highlight a small number of nodes in the tree – one of which being the node of interest. Then they can go through these highlighted nodes until they have found the version they were looking for, either by switching to them by clicking, or by moving their cursor over the nodes and using the modification summary feature.

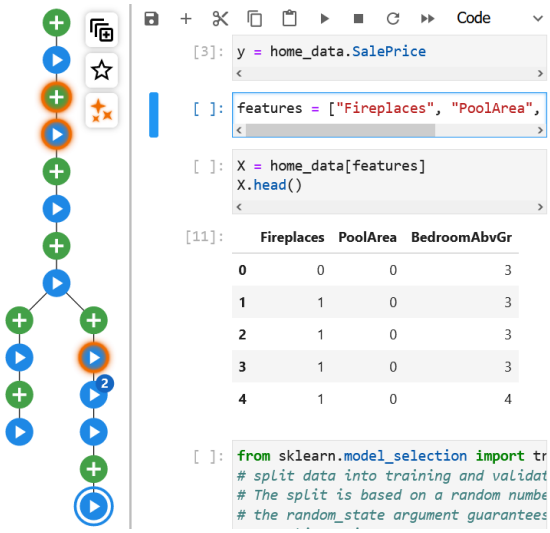

Fig. 5: The relevant node highlighting feature in use

B. Summary

We have selected the features of HISTREE to support data scientists performing multiple experiments in non-linear workflows. We have made an effort to ensure that all user requirements listed in Section I are at least partially addressed. Table II shows a course-grained mapping between those requirements and the implemented features.

VI. USER EXPERIMENTS

To collect feedback from data scientists and gain insights into how they use HISTREE for solving ML problems, user experiments were designed and conducted, guided by the Technology Acceptance Model (TAM) [16]. It defines *usefulness* and *ease of use* as central factors for acceptance:

TABLE II: User requirements covered by HISTREE

Features	User Requirements R1	R2	R3
Switching Notebook Versions (F1)			•
Branching (F2)		•	
Modification Summary (F3)	•		•
Bookmarking (F4)	•		
Relevant Node Highlighting (F5)			•

- *Usefulness* refers to the perceived probability that a tool will increase job performance. In our case, we want to investigate to what extent HISTREE supports organizing the conducted experiments.
- *Ease of use* refers to the extent to which the user perceives the usage of a tool, i.e. HISTREE, as effortless.

In the following, we will describe the design and process of the user experiments.

A. Participants

Ten participants, P01 to P10, took part in the user experiments. All participants were academic data scientists who gained their experience through data science and ML-related university courses or private projects.

All participants were asked for a self-assessment of their experience with regard to Python and Jupyter Notebooks on a continuous scale between 1 (very little experience) and 5 (a lot of experience). The results of the self-assessments can be found in Table III. The average experience for Python and Jupyter Notebooks was graded 3.1 and 3, respectively. The highest grade of experience was 4.5 for Python, and 5 for Jupyter Notebooks, while the lowest grade of experience was 2 for both categories. Therefore, the participants represent a wide range of experience with Python and Jupyter Notebooks.

None of the participants had any experience with history tracking extensions like VERDANT in the Jupyter Notebook environment. However, five out of the ten participants had used Git for version control purposes with Jupyter Notebooks.

All participants took part in the study voluntarily and gave their consent to the recording of the interview.

TABLE III: Participants' self-assessed experience

	Python Exp.	JN Exp.	Used JN w/ Git
P01	3	3	Yes
P02	2	2	No
P03	2	3	No
P04	2	3	No
P05	4.5	4	No
P06	4	2	Yes
P07	3	2	Yes
P08	4.5	5	Yes
P09	3	3	No
P10	3	3	Yes
Average	3.1	3	–

B. Task

Finding a suitable task to give to the participants for evaluating HISTREE was a major challenge. If the task was too simple, it may not warrant a lot of experimentation from the participants and the features of HISTREE – branching in particular – would not be used very much. A certain level of complexity is also needed to reflect real-world tasks that data scientists are performing daily. However, the given task cannot be too complex, as the evaluation would take too long to complete and may also be too difficult for some of the participants to perform properly without step-by-step guidance.

The selected task was about constructing a model for predicting housing prices using a popular data set from the Kaggle Intro to Machine Learning tutorial [17].

C. Procedure

We structured the user experiments into three parts, which are described below. They were held in German, except for the one with P10 which was held in English. The user experiments lasted between 40 and 85 minutes with an average of about 58 minutes and were conducted in a semi-structured format.

Part 1 - Getting to Know HISTREE*:* In the first part, the participants were shown a Jupyter Notebook in the JupyterLab environment with the HISTREE extension open in the sidebar. They were also given a very brief explanation of HISTREE.

They were then asked to play around with the HISTREE extension to get used to the concept of the experiment history tree and explore the user interface. No proper ML-related task was given out during this part.

Part 2 - Solving an ML Task: Next, a Notebook containing two different solutions to the chosen housing prices problem was shown to the participants. Both solutions, each using a different regression model, were situated in a different branch of the experiment history tree. The participants were informed that this Notebook was in a ”cold” state, meaning that it was opened with a fresh kernel without any variables being filled yet.

First, the participants were asked to find out how well the two already existing solutions have performed, i.e., how high the average validation error was for the models.

After that, they were asked to create a better model for predicting housing prices. As part of the task, the participants were encouraged to try out a model other than the ones already used in the Notebook. The participants were told to use the features of HISTREE whenever they believed them to be useful. But they were allowed to use any other tools they would usually use for completing this kind of task, like searching the internet. They were also allowed to ask the experiment lead data science-related questions.

Part 3 - Answering Questions and Collecting Feedback: After the participants completed this task to a degree that they were satisfied with, they were asked questions to find out how they assess the HISTREE extension and what they wanted to see improved.

The following five questions regarding the usefulness of the HISTREE features were asked, all being answered with values on a continuous scale between 1 (not useful) and 5 (very useful).

- Q1: How useful is the switching Notebook versions feature (F1), i.e. to look at previous versions using the experiment history tree?
- Q2: How useful is the branching feature (F2), i.e. to create branches in the experiment history tree?
- Q3: How useful is the modification summary feature (F3), i.e. to see a summary of the changes that occur in a node by hovering over it?
- Q4: How useful is the relevant node highlighting feature (F4), i.e. to see which nodes have an impact on the selected cell?
- Q5: How useful is the bookmarking feature (F5), i.e. to mark important nodes in the tree?

Furthermore, we wanted to get an overall score from the participants. There, two more questions were asked.

- Q6: Does HISTREE provide too much, too little, or exactly the right amount of information?
- Q7: How likely are you to use HISTREE for further projects?

For Q7, the likelihood was asked to be rated as values on a continuous scale between 1 (not likely) to 5 (very likely). At the end, the participants were asked for any further feedback and suggestions to further assess the ease of use.

VII. ANALYSIS OF THE RESULTS

The following section presents the results obtained in the user experiments and their interpretation, a discussion of threats to validity, and a final summary.

A. Usefulness of the Features

First, we look at the participants' scores on the usefulness of the HISTREE extension (questions Q1 to Q5), interpret the given scores, and sum up the opinions of the participants with regard to each feature. The average scores are visualized in Figure 6, and the values given by the participants can be seen in Table IV.

TABLE IV: Responses given to questions Q1-Q7

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
P01	5	5	5	5	5	Just Right	5
P02	4	5	5	4	2	Just Right	4.5
P03	5	5	5	5	4	Just Right	5
P04	5	4.5	4	3	3	Too Much	5
P05	5	5	5	4	2	Too Much	5
P06	5	5	5	4	3	Too Much	3
P07	5	4.5	4.5	5	4	Just Right	5
P08	4.5	5	4	5	3	Too Much	4
P09	5	5	4	4	3	Just Right	5
P10	5	5	3.5	4	3	Just Right	4

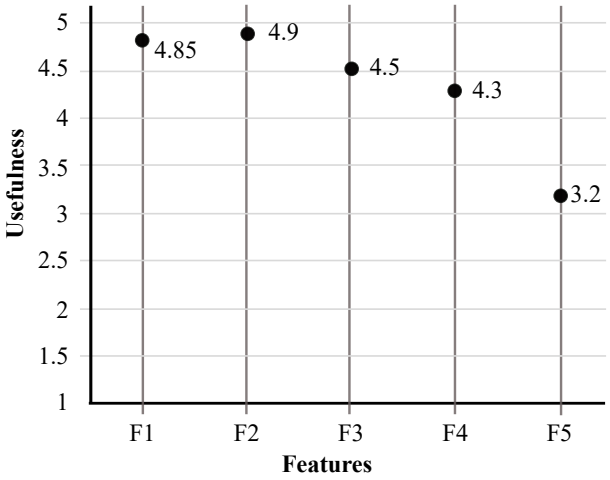

Fig. 6: Average values for features F1-F5

Switching Notebook Versions (F1): The ability to switch between different Notebook versions was rated to be highly useful by the participants, responding to Q1 with an average grade of 4.85. Eight out of the ten participants gave full marks. This feature was used extensively when solving the given problem, mostly to look at the solutions in the other branches, and occasionally copy code from other branches to their own branch. It was also the primary feature used for finding specific information from the history of the Notebook, like when looking up how much error the models from the different branches produced.

Branching (F2): The branching feature achieved the highest score among the assessed features, receiving an average grade of 4.9 for Q2, with eight participants giving it full marks, and the remaining two scoring it with a 4.5. Participants P01 and P07 said that this feature helped to compare the different approaches by helping to organize the experiments. Participant P07 also added that the tree representation much better supports grasping the history of the experiments as opposed to having just one linear timeline of changes. The participants were quickly able to create a new branch and were using branching effectively in their experimentation efforts. About half of the participants created only one additional branch for the new solution and stuck to it without further branching out. The other half also used further smaller subbranches for backtracking when an experiment failed, or to try different features or a model if they had not already done so in their initial branch.

Modification Summary (F3): The modification summary feature was well regarded by the participants, receiving an average grade of 4.5. It was considered so important to participants P03 and P06 that they said that the HISTREE extension would not make a lot of sense without this feature. The modification summary was mostly used for quickly finding out what changed in a Notebook version or a small set of versions. This was often done for the purpose of finding a particular version the participants wanted to look at more closely or branch out of.

A major point of criticism was the way the pinning of the modification summary was handled. Some participants had trouble finding the pin button, and most wanted to move the mouse over to the modification summary without clicking the pin button, expecting the modification summary to stay open.

Highlighting Relevant Nodes (F4): Once the participants understood the idea behind the relevant node highlighting feature, they found it to be very useful in certain situations. The average grade for the feature was 4.3. P01 and P10 said that it would have been more useful in a more complex project. The generally positive feedback is somewhat surprising, as there were not many situations in the study where the relevant node highlighting would have been of great benefit due to the brevity of the given task. Still, it was sometimes used to find different versions of a particular cell, or a specific version of the Notebook where they knew they edited a particular cell. P05 and P07 remarked that the feature helped manage the size of the tree, as it made it easier to see the parts of the tree that they were interested in at the moment. Even in cases where it does not provide any use, it was noted by P04 that it does not detract from the extension, as it is fully optional.

Bookmarking (F5): The bookmarking feature was not as well received by the participants, receiving an average grade of 3.2 in response to Q5. It was still regarded as potentially useful by most participants, but often with the caveat that the data scientist should have the ability to label their bookmarks. Participants P05, P06, and P09 did not see much use in the feature, saying that they already have a good understanding of where everything is situated within the tree. Given these mixed results, it is not surprising that bookmarks were rarely used during the experiments.

B. General Acceptance

At the end of the user experiments, the participants were asked to assess the sufficiency of the information provided by HISTREE and whether HISTREE is a tool they would want to use in the future. The individual categorical and quantitative answers can be seen in Table IV.

Amount of Provided Information (Q6): The consensus among participants about the amount of information provided by HISTREE can be summed up as generally adequate, but leaning towards too much. Out of the total ten participants, six expressed that the information was *just right*, while four felt that it was *too much*. The main complaint was that the tree could grow too big, which was mentioned by half of the participants (P04 - P08). To alleviate this problem, participants P04 and P08 suggested a more flexible collapsing system that would enable the data scientist to collapse any group of nodes within the tree and not just sub-trees, as is currently the case with HISTREE. This would make the size of the tree more manageable, as long as the data scientist occasionally collapses sequences of nodes that they are not interested in anymore.

Future Usage (Q7): On a continuous scale from 1 (not likely) to 5 (very likely), the average score obtained was 4.55, indicating a strong inclination towards using HISTREE in future projects. Nearly all participants expressed their desire to incorporate the HISTREE extension, with only one participant scoring below 4 in terms of likelihood.

According to P02, even if the project does not immediately call for intense usage of features like branching, it would still be good to have the option of using these later on. The same participant also saw a lot of value in the automatic versioning aspect of the extension, calling it "Git for free". Participant P05 even said that they could imagine this extension as a standard feature in JupyterLab. Only P06 was not yet fully convinced that they would try it again, stating that it was not because of HISTREE itself, but rather because they dislike the fundamental concept of Jupyter Notebooks. In fact, they thought so highly of the features of HISTREE that they subsequently asked if they could also be implemented in a more traditional code editor.

C. Threats to Validity

Internal Validity: The user experiments were conducted in person with the same hardware for each participant, except for P10 who accessed a computer running with the extension using a remote access software. The impact of this is limited, as no significant problems occurred due to the remote nature of the interview compared to the other participants.

A more significant threat would be that the participants knew that the purpose of the experiments was for evaluating HISTREE, presumably using it more intensely or adjusting their usual workflow for the task, which they would not have done otherwise.

External Validity: One threat to the external validity of the study is the fact that all participants were of a relatively narrow demographic, namely university members. While they do represent a broad range of skill levels and some have significant experience with Python, Jupyter Notebooks, and data science, there were no professional data scientists among the participants. The generalizability of the results is also somewhat hampered by the relatively small sample size of ten individuals.

Due to the time constraints of the interviews, as well as the large variety of skill levels among the participants, the task had to be relatively simple. While it was chosen with great care to have enough complexity to give a reason to use the features of HISTREE, it does not represent a larger real-world ML experimentation effort.

The short amount of time the participants could work with HISTREE also means that longer-term use over the course of multiple days or weeks could not be assessed. It should also be stressed that this was the first time for each participant to use HISTREE. Thus, no concrete conclusions can be drawn about how someone who has had more time with the extension would use it.

VIII. DISCUSSION

In our user experiments, the HISTREE extension was able to fulfill the requirements set out at the beginning, with the experiment history tree offering a capable way of both visualizing (R1) and traversing (R3) the experiments conducted in a Jupyter Notebook. Its automatic versioning capabilities made the tracking of experiments and their results practically effortless (R2, R4) while offering a good amount of information to the data scientist (R5). The only major problems identified were that the experiment history tree could potentially grow too large, especially for larger projects, and that bookmarks cannot be labeled. Overall, the participants were highly impressed with the extension and its features. Going back to the Technology Acceptance Model [16], HISTREE proved to be both useful and easy to use. The participants were able to make good use of the extension's feature set during the task, keeping their experimentation efforts organized in branches, which they frequently switched between to compare their approaches and findings. They seemed to enjoy using the extension and did not have significant trouble understanding how it functions or how to use it.

The positive attitude towards HISTREE gives great confidence in the viability of the extension in real-world usage.

IX. CONCLUSION & FUTURE WORK

In this paper, we have presented a tree-based automated approach to organizing ML experiments in Jupyter Notebooks. We have made the following contributions:

- We have identified user and usability requirements for tools used by data scientists for their experiments and have highlighted the shortcomings of the already-known tools (Sections I & II).
- We have developed a tree-based experiment history model to adequately represent the non-linear experiments performed, which is technically based on the concept of Notebook versions (Section III).
- We have identified the relevant version-forming operations in Notebooks, and mapped those to the experiment history model (Section IV).
- We have developed a tool, the JupyterLab extension HISTREE, that implements the experiment history model and provides important features that support data scientists in conducting the experiments (Section V).
- We have conducted and analyzed user experiments according to the Technology Acceptance Model to evaluate the usefulness and ease of use of HISTREE (Sections VI & VII).

Besides the overall positive feedback, the evaluation also highlighted some parts of HISTREE that should be improved for future versions. Among many smaller tweaks to the extension, bookmark labeling as well as a more flexible way of collapsing groups of nodes was requested in particular, with the latter being important for combating the issue of the tree becoming too large over time. While HISTREE does offer a way of collapsing sub-trees, this was deemed insufficient by

the participants who asked for any group of consecutive inner nodes in the experiment history tree to be collapsible.

An important and widespread problem not yet properly covered in this paper is the poor reproducibility that Jupyter Notebooks generally suffer from, mostly due to problematic dependency management and non-linear execution order [5]. HISTREE does have the potential of mitigating the latter issue, as it offers a way of comprehending the execution history of the Notebook in detail using the experiment history tree. Evaluating to which extent this can improve reproducibility is subject to future research. While it is not currently part of the feature set, a way of tracking the packages used in a Notebook could also potentially be added to HISTREE to help with recreating the original computing environment to improve reproducibility.

After the tool is deployed and publicly available, future research should focus on surveys and/or studies with data scientists who have tried out the extension over a longer period and for larger projects. Such studies could aid significantly in further improving HISTREE to become even more useful to data scientists in their day-to-day work.

Complementary materials for reviewers:
Data of user experiments:
https://anonymous.4open.science/r/HistreeBlinded-BE0F
Demonstration video of HISTREE (2:47):
https://youtu.be/lCylbZacPuw

REFERENCES

[1] Kery, Mary Beth and Myers, Brad A, "Exploring exploratory programming," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2017, pp. 25–29.

[2] Davidson, Susan B. and Freire, Juliana, "Provenance and scientific workflows: Challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1345–1350. [Online]. Available: https://doi.org/10.1145/1376616.1376772

[3] Jeffrey M. Perkel, "Why jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7729, pp. 145–146, Oct. 2018. [Online]. Available: https://doi.org/10.1038/d41586-018-07196-1

[4] Zhang, Amy X. and Muller, Michael and Wang, Dakuo, "How Do Data Science Workers Collaborate? Roles, Workflows, and Tools," *Proc. ACM Hum.-Comput. Interact.*, vol. 4, no. CSCW1, may 2020.

[5] Pimentel, João Felipe and Murta, Leonardo and Braganholo, Vanessa and Freire, Juliana, "A large-scale study about quality and reproducibility of jupyter notebooks," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 507–517.

[6] Kery, Mary Beth and Radensky, Marissa and Arya, Mahima and John, Bonnie E. and Myers, Brad A., "The story in the notebook: Exploratory data science using a literate programming tool," ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–11. [Online]. Available: https://doi.org/10.1145/3173574.3173748

[7] Ramasamy, Dhivyabharathi and Sarasua, Cristina and Bacchelli, Alberto and Bernstein, Abraham, "Visualising data science workflows to support third-party notebook comprehension: An empirical study," *Empirical Softw. Engg.*, vol. 28, no. 3, mar 2023. [Online]. Available: https://doi.org/10.1007/s10664-023-10289-9

[8] Subramanian, Krishna and Hamdan, Nur and Borchers, Jan, "Casual notebooks and rigid scripts: Understanding data science programming," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2020, pp. 1–5.

[9] Rule, Adam and Tabard, Aurélien and Hollan, James D, "Exploration and explanation in computational notebooks," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[10] Subramanian, Krishna and Maas, Johannes and Borchers, Jan, "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.

[11] Kery, Mary Beth and Horvath, Amber and Myers, Brad A, "Variolite: Supporting Exploratory Programming by Data Scientists." in *CHI*, vol. 10, 2017, pp. 3–025.

[12] Kery, Mary Beth and Myers, Brad A., "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.

[13] Kery, Mary Beth and John, Bonnie E. and O'Flaherty, Patrick and Horvath, Amber and Myers, Brad A., "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. [Online]. Available: https://doi.org/10.1145/3290605.3300322

[14] Nextjournal GmbH, "The notebook for reproducible research — Nextjournal," https://nextjournal.com/home, 2021, [Accessed 13-Jun-2023].

[15] Liu, Yang and Kale, Alex and Althoff, Tim and Heer, Jeffrey, "Boba: Authoring and visualizing multiverse analyses," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1753–1763, 2021.

[16] Riemenschneider, Cynthia K and Hardgrave, Bill C, "Explaining software development tool use with the technology acceptance model," *Journal of Computer Information Systems*, vol. 41, no. 4, pp. 1–8, 2001.

[17] Kaggle, Inc., "Intro to Machine Learning," https://www.kaggle.com/learn/intro-to-machine-learning, accessed: 2023-04-07.