PROXAI: Interactive Provenance-Aware Debugging of Machine Learning Pipelines

Pasquale Leonardo Lazzaro
Università Roma Tre
Italy
pasqualeleonardo.lazzaro@uniroma3.it

Elia Guglielmi
Università Roma Tre
Italy
eli.guglielmi1@stud.uniroma3.it

Paolo Missier
University of Birmingham
United Kingdom
p.missier@bham.ac.uk

Riccardo Torlone
Università Roma Tre
Italy
riccardo.torlone@uniroma3.it

ABSTRACT

Data provenance is a well-established foundation for transparency and auditability in data management, yet existing provenance systems offer little support for connecting lineage information to downstream model behavior. Conversely, Explainable AI methods identify influential features and training instances, but treat the training data as a fixed input, leaving their preprocessing history unexamined. We present **PROXAI**, an interactive system that bridges these two perspectives to support provenance-aware debugging of machine learning pipelines. Starting from anomalous model behavior, **PROXAI** uses XAI techniques to identify influential data elements and follows their lineage through a multi-granular provenance graph, enabling users to trace preprocessing root causes, inspect transformation logic, and validate corrective actions within a unified debugging workflow. Our demonstration highlights how combining XAI and provenance enables end-to-end explanation and practical debugging of data-centric AI pipelines.

PVLDB Reference Format:
Pasquale Leonardo Lazzaro, Elia Guglielmi, Paolo Missier, and Riccardo Torlone. **PROXAI**: Interactive Provenance-Aware Debugging of Machine Learning Pipelines. PVLDB, 19(12): 4706 - 4709, 2026.
doi:10.14778/3827998.3828102

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/pasqualeleonardolazzaro/PROXAI.

1 INTRODUCTION

The reliability of machine learning systems depends not only on model design, but also on the quality and processing history of the data used for training. While Explainable AI (XAI) has produced effective methods for interpreting model decisions and identifying influential training instances, these methods typically treat the training data as a fixed input and provide little insight into how such instances were generated. Conversely, data provenance has long been studied as a foundation for reproducibility, transparency, and auditability in data processing pipelines [3, 8]. While provenance techniques accurately reconstruct data transformations [7], they usually stop at the model training boundary, offering limited support for linking lineage to specific model outcomes. As a result, when a model exhibits problematic behavior, it remains difficult to trace that behavior back to the preprocessing steps that introduced the issue.

These two lines of work are naturally complementary. XAI can identify which training instances most strongly affect a prediction, but not why those instances appear in their current form. Fine-grained provenance, in turn, can explain how data was produced, but without a targeted entry point, it is often difficult to use effectively for model debugging. Combining these perspectives enables a more actionable debugging workflow, in which XAI provides the focus and provenance provides the context needed to localize the root cause of model failures.

Building on this observation, we present **PROXAI** (PROvenance for eXplainable AI), an interactive system for provenance-aware debugging of machine learning pipelines. Starting from a problematic prediction or anomalous model behavior, **PROXAI** uses influence-based interpretability techniques to identify the training instances that most affect the prediction and follows their lineage through a multi-granular provenance graph to recover the transformations that generated them. This allows users to move from model-level signals to data-level lineage and inspect the operations that shaped the training data responsible for the observed behavior. To support this analysis, **PROXAI** adopts a unified graph-based representation explicitly connecting data preparation, training instances, and model behavior. The system tracks provenance at multiple levels of granularity, from datasets to rows and values, enabling influential training instances to be traced back to the preprocessing steps that produced them. This design draws on our earlier work on fine-grained provenance collection [2, 5], but repurposes it within a broader framework centered on the interaction between data lineage and model behavior. **PROXAI** also supports interactive exploration of lineage paths, provenance metadata, and transformation logic, enabling users to inspect the evidence behind a diagnosis and validate corrective actions within an end-to-end debugging loop.Thus, the demo shows how anomalous behavior can be traced back to specific preprocessing operations, turning model-level signals into pipeline-level explanations.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828102

This paper makes three contributions. First, it presents an interactive framework that connects prediction-level explanations with fine-grained data provenance for end-to-end debugging of ML pipelines. Second, it introduces a multi-granular provenance workflow that traces influential training instances back to the preprocessing operations that generated them. Third, it demonstrates, through a hands-on interactive session, how this integration supports practical root-cause analysis and corrective validation across representative debugging scenarios.

2 OVERVIEW OF PROXAI

PROXAI combines influence-based model interpretability and fine-grained provenance analysis to support provenance-aware debugging of ML pipelines. Its main features are described below.

2.1 XAI-Guided Debugging Entry Points

To connect model behavior with upstream data preparation, **PROXAI** integrates an interpretability module that identifies the data elements most relevant to a prediction. Rather than treating explanations as an end product, the system uses them as targeted entry points for provenance analysis. This is achieved through influence analysis at two complementary levels:

Feature-level analysis. The system uses SHAP (SHapley Additive exPlanations) [6] to derive a global importance ranking over features by aggregating local explanations across a set of predictions. The top-ranked columns can then be traced through schema-level provenance across their transformation history.

Instance-level analysis. The system supports instance-wise influence methods that identify the training samples with the strongest impact on a given prediction. The current implementation includes Influence Functions [4] and TracIn [1], returning the top-k influential training instances for the current prediction.

The identified features and training instances are mapped to the provenance graph, thereby focusing the subsequent analysis on the data elements most relevant to the observed behavior.

2.2 Interactive Provenance Exploration

Once relevant data elements have been identified, **PROXAI** supports interactive exploration of their lineage through a graph-based provenance representation. The graph extends W3C PROV to capture entities, transformations, and column structures, enabling users to trace provenance across multiple granularities (from datasets down to values).

To support inspection and querying, the graph is enriched with semantic metadata extracted from the pipeline code, including logical structuring of scripts and concise descriptions of transformation steps. On top of this representation, **PROXAI** supports both direct graph exploration and natural-language interaction. In the latter case, the system relies on a retrieval-augmented workflow that combines structured graph querying with hybrid retrieval over provenance metadata, leveraging both semantic and structural embeddings to support flexible access to lineage paths, transformation logic, and code-level operations.

These capabilities turn the provenance graph into an interactive debugging substrate rather than a passive lineage log, enabling users to inspect both structural and semantic evidence behind a diagnosis.

2.3 Connecting Model Explanations and Provenance

The key contribution of **PROXAI** lies in the tight integration of these two components. Influence-based explanations identify the training data that matters most for a problematic prediction, while provenance analysis explains how that data was produced. By using XAI outputs to guide provenance retrieval, **PROXAI** concentrates the analysis on the most relevant subset of entities, attributes, and transformations.

Without influence-based guidance, provenance analysis is difficult to use effectively for debugging because of the size and complexity of the graph. Without provenance, explanations remain incomplete because they do not reveal which preprocessing operations created the relevant data points. Together, these components enable an end-to-end explanatory workflow from model-level signals to pipeline-level root causes and corrective validation.

3 SYSTEM DESCRIPTION

3.1 System Architecture

Figure 1 shows the logical architecture of **PROXAI**, organized into four macro-phases: *Pipeline Understanding*, *Execution and Provenance Capture*, *Direct Analysis*, and *Natural-Language Analysis*. Beyond identifying the main system components, the numbered blocks also describe the workflow through which **PROXAI** supports end-to-end provenance-aware debugging.

The workflow starts with *Script rewriting* (❶), where the input script is annotated, segmented into logical activities, and analyzed to identify the involved columns. This phase also produces an *Internal dictionary*, which stores operation descriptions and semantic metadata that later enrich provenance information. The rewritten script is then executed in its native *Execution environment* (❷) over the input dataset.

During execution, the *Provenance generator* (❸) compares the inputs and outputs of each operation, constructs provenance fragments, and injects the semantic descriptions stored in the internal dictionary. At the same time, execution produces the preprocessed dataset used to train and test the *ML Model*. The resulting provenance fragments are materialized in the provenance graph, which acts as the central substrate for subsequent analysis.

Once the model has been trained and tested, the *XAI Generation* module computes feature- and instance-level explanations through SHAP, Influence Functions, and TracIn. These influential points are passed to the *End-to-end explanation* component (❹), which combines explanation signals with provenance data. The technical novelty of this component lies in resolving the non-trivial alignment between multi-granular XAI outputs (e.g., local instance-level attributions vs. global feature importance) and the multi-hop relational structure of data lineage, effectively connecting model behavior with the specific upstream transformations that produced the relevant training data.

From this point, the workflow can follow two complementary analysis paths. Through *Direct access & visualization* (❺), users can

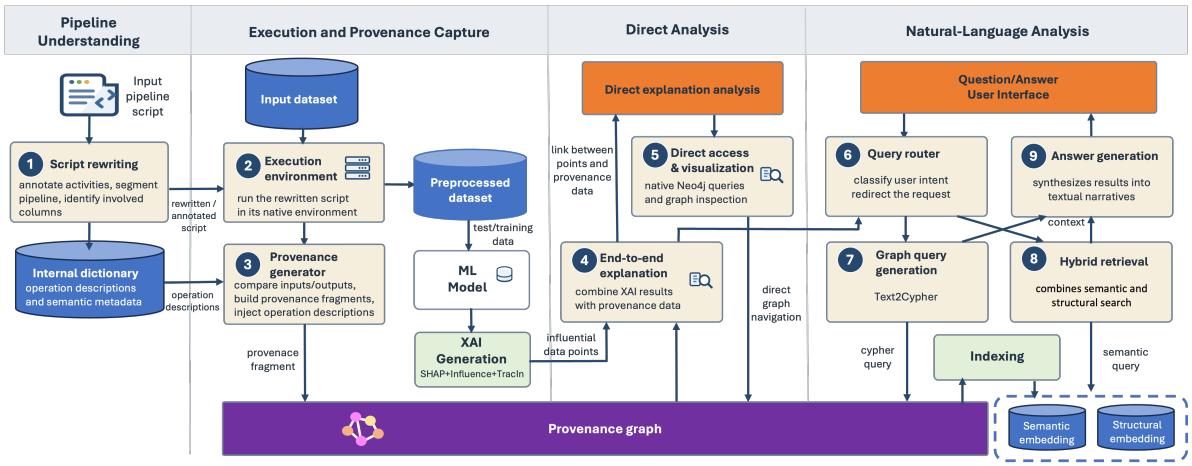

Figure 1: PROXAI architecture

inspect the graph natively and visually explore the links between influential points and provenance entities, directly analyzing lineage and transformation structure.

Since manually exploring large and complex provenance graphs can impose a significant cognitive burden and manual effort on users, requests can alternatively be handled through the natural-language interface. In this case, the *Query router* (❻) classifies user intent and redirects the request either to *Graph query generation* (❼), where questions are translated into Cypher queries, or to *Hybrid retrieval* (❽), which combines semantic and structural search over indexed provenance information. Following a retrieval-augmented generation workflow, the retrieved context is then processed by the *Answer generation* component (❾), which synthesizes the results into textual answers returned through the question-answer user interface.

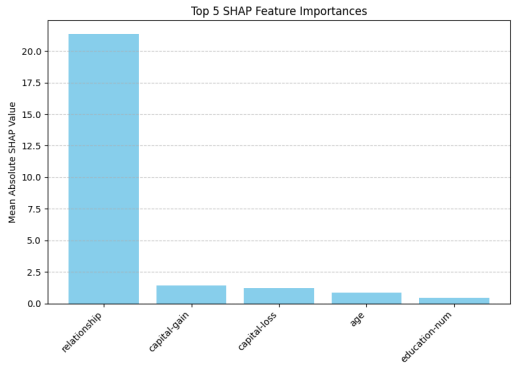

Figure 2: SHAP-based feature importance

3.2 PROXAI in Action

We illustrate the use of **PROXAI** through a deliberately simple debugging scenario based on the Census Income dataset, chosen to clearly expose the main steps of the workflow. A developer preprocesses the data and trains a simple neural network that achieves near-perfect training accuracy (99.8%) but only 30.75% test accuracy. This gap suggests that a serious issue has been introduced in the preprocessing pipeline, which includes 14 feature transformations.

Rather than manually inspecting the entire pipeline, the developer starts from the XAI module. A global SHAP analysis over the test set reveals that the model relies disproportionately on the relationship feature, whose importance dominates the remaining attributes by a large margin, as shown in Figure 2. This abnormal pattern suggests that the feature may be affected by target leakage.

Starting from this signal, **PROXAI** allows the user to move directly from the SHAP results to the corresponding node in the provenance graph, as shown in Figure 3, and then inspect the transformation history of the suspicious feature through the interactive querying interface.

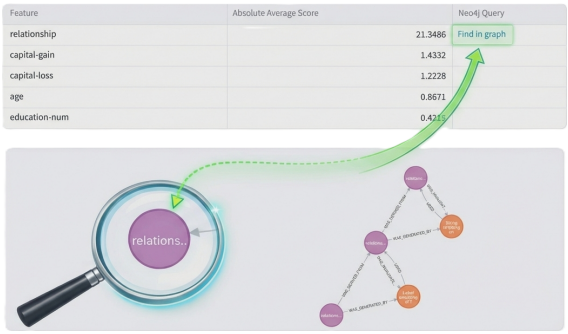

Figure 3: Interactive connection between SHAP results and provenance graph visualization

For example, querying *"What is the code of activity 'Label encoding of the relationship column'?"* retrieves the corresponding operation and source code (Figure 4).

More generally, the interface supports conversational inspection of the suspicious transformation and its role in the pipeline. In this case, the returned code reveals the origin of the problem:

```
Erroneous code causing target leakage
relationship_trans = LabelEncoder()
df["relationship"] = relationship_trans.fit_transform(
    df["label"].values.astype(str))  # Should be
    ↪ df["relationship"]
```

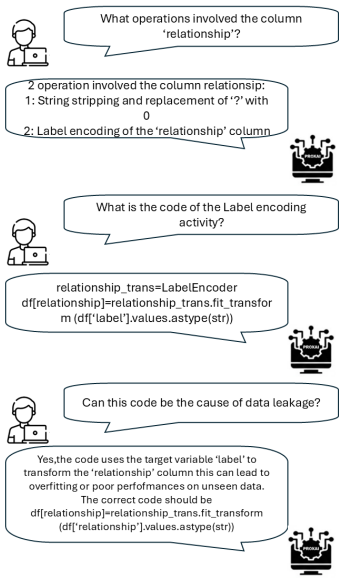

Figure 4: Interactive query over provenance metadata

The issue is that relationship was encoded using the target variable label rather than the original column, introducing a direct leakage path. After correcting this line, the model's test accuracy increases from 30.75% to 82.54%, an improvement of 51.79 percentage points.

4 DEMO OUTLINE

The demonstration is organized as a guided debugging session over a preloaded ML pipeline exhibiting anomalous behavior due to a data preparation issue. The goal is to show how **PROXAI** turns model explanations into an end-to-end workflow by executing the following steps:

Step 1: From anomalous model behavior to influential data. The demo begins from the observed behavior of the trained model, whose mismatch between training and test performance suggests the presence of a serious issue in the preprocessing pipeline. Attendees first inspect the output of the XAI module, which provides both feature-level and instance-level explanations for the current model behavior. Depending on the selected scenario, **PROXAI** can highlight suspicious features through SHAP-based importance analysis or identify the training samples that exert the strongest positive or negative influence on a given prediction. This first step shows the role of XAI as a focused entry point for debugging. Rather than stopping at the explanation itself, **PROXAI** uses these signals to identify the subset of data elements that should be examined more closely in the provenance graph.

Step 2: From influential data to provenance and transformation logic. Starting from the suspicious features or training instances identified in the previous step, attendees then explore their provenance through **PROXAI**. The system allows users to move directly from the XAI outputs to the corresponding entities in the provenance graph, thereby connecting model-level explanations with data-level lineage. At this stage, attendees can pursue the analysis through direct graph exploration or through interactive querying over provenance metadata, depending on the selected scenario and on the aspects they wish to inspect more closely. This makes it possible to recover lineage paths, examine the sequence of preprocessing activities that affected a feature or record, and inspect the code associated with a given operation. In this way, the demo shows how anomalous model behavior can be traced back to specific preprocessing operations and their associated logic, turning model-level signals into pipeline-level explanations.

Step 3: From diagnosis to corrective validation. In the final step, the faulty transformation is corrected and the pipeline is executed again. **PROXAI** then allows attendees to compare the updated model behavior, the new explanation outputs, and the revised provenance information with the original faulty execution. This final stage closes the explanatory loop. Once the problematic transformation is fixed, the model behavior becomes more plausible, the anomalous explanatory signals disappear or are reduced, and predictive performance improves. The demo therefore shows not only how **PROXAI** supports root-cause analysis, but also how it enables users to validate that the diagnosed cause was indeed responsible for the observed behavior.

5 CONCLUSION

PROXAI demonstrates how combining model explanations and fine-grained provenance supports interactive debugging of ML pipelines. By connecting anomalous model behavior directly to upstream data transformations, the system enables practical root-cause analysis and corrective validation within a unified, end-to-end explanatory workflow.

REFERENCES

[1] Pruthi Garima, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Proc. of NIPS*. Article 1672.

[2] Luca Gregori, Pasquale Leonardo Lazzaro, Marialaura Lazzaro, Paolo Missier, and Riccardo Torlone. 2025. An LLM-guided platform for multi-granular collection and management of data provenance. *J. Big Data* 12, 1 (2025), 187. doi:10.1186/S40537-025-01209-3

[3] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. A survey on provenance: What for? What form? What from? *The VLDB Journal* 26, 6 (2017), 881–906. doi:10.1007/S00778-017-0486-1

[4] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Proc. of 34th ICLM*, Vol. 70. PMLR, 1885–1894.

[5] Pasquale Leonardo Lazzaro, Marialaura Lazzaro, Paolo Missier, and Riccardo Torlone. 2025. PROLIT: Supporting the Transparency of Data Preparation Pipelines through Narratives over Data Provenance. In *Proc. of EDBT*. OpenProceedings.org. doi:10.48786/edbt.2025.336

[6] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proc. of NIPS*. 4768–4777.

[7] Mohammad Hossein Namaki, Avrilia Floratou, Fotis Psallidas, Subru Krishnan, Ashvin Agrawal, Yinghui Wu, Yiwen Zhu, and Markus Weimer. 2020. Vamsa: Automated Provenance Tracking in Data Science Scripts. In *Proc. of KDD*. ACM, 1542–1551. doi:10.1145/3394486.3403205

[8] João Felipe Pimentel, Juliana Freire, Leonardo Murta, and Vanessa Braganholo. 2019. A survey on collecting, managing, and analyzing provenance from scripts. *Comput. Surveys* 52, 3 (2019), 1–38. doi:10.1145/3311955