From XAI to XEE: eXplainable End-to-End using Influence and Provenance

Paolo Missier[1], Riccardo Torlone[2,*], Pasquale Leonardo Lazzaro[2], Luca Gregori[2], Clyde Fernandes[1] and TianYang Xie[1]

[1]University of Birmingham, Edgbaston, UK
[2]RomaTre University, Rome, Italy

Abstract

It is generally accepted that Machine Learning and AI models can be made "trustworthy" by providing explanations that justify their predictions, and multiple techniques have been proposed to achieve this. These are collectively referred to as model-based explanations, and the kind of models that can be explained using those are referred to as XAI. There is, however, also a complementary need to provide *data explanations* that account for the dataset selection and data engineering steps that precede model training. Data explanations can be obtained by capturing and then examining the provenance of the data in a training set. In this position paper, we suggest that there is value in combining these two types of explanations, producing an "eXplainable End-to-End" (XEE) backward path that originates from model predictions and traces all the way back to raw data, using a combination of XAI and provenance methods. We demonstrate how this can be achieved in practice using Influence Functions, a well-known XAI technique, in combination with PROLIT, a provenance-based tool that we have been developing to provide data explanations. We provide a simple but representative example and suggest the next steps for more in-depth research into XEE.

Keywords

Data-Centric AI, Explainable AI (XAI), Data Provenance, Influence Functions, End-to-End Explainability

1. Introduction

Multiple methods are available to achieve transparency and explainability of predictions made by Machine Learning and AI models [1], including highly nonlinear, complex models such as Deep Neural Networks, which are generally regarded as "black boxes" from an interpretability perspective. In such complex settings, a specific model outcome can be "explained" by providing examples of data points in the training set, which have been the *most influential* contributors to the outcome. We rely on Influence Functions [2, 3] to retrieve training points that contribute to specific predictions. These are summarized in Section 2.

In this paper, we explore and illustrate the idea of combining model explanations with *data explanations* to provide an end-to-end account of a model's prediction. More precisely, consider

SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16–19, 2025, Ischia, Italy
*Corresponding author.
p.missier@bham.ac.uk (P. Missier); riccardo.torlone@uniroma3.it (R. Torlone); pasqualeleonardo.lazzaro@uniroma3.it (P. L. Lazzaro); luca.gregori@uniroma3.it (L. Gregori); caf156@student.bham.ac.uk (C. Fernandes); txx400@student.bham.ac.uk (T. Xie)
0000-0002-0877-7063 (P. Missier); 0000-0001-7116-9338 (R. Torlone); 0009-0002-9489-4916 (L. Gregori)

a training set that was constructed from a collection of initial datasets that are processed using a workflow consisting of data transformation operators.

The *data explanation* of data point d in the training set is d's provenance, for which we adopt the W3C PROV model (http://www.w3.org/TR/prov-dm/) and its standard graph representation

Such end-to-end explanations are useful in many practical settings. For instance, a data preparation pipeline may include data transformations that may affect model performance by accidentally injecting out-of-distribution data points or by altering the set through data augmentation or other forms of synthetic data injection. In such cases, it is important to identify the influential data points and trace them back through the pipeline to understand where they come from.

Given a model $h : \mathbb{R}^n \rightarrow \mathbb{R}$ trained on D, model explanations and data explanations can be described in abstract terms using two functions. The first is a function I that maps the outcome $\hat{y} = h(\mathbf{x})$ for input $\mathbf{x}$ to a set of influential training data points in the training set D:

$$\mathcal{I}(\hat{y}) = \{d_1 \dots d_k\}, d_i \in D$$

The second returns the provenance $prov(d)$ of a training point d.

Conceptually, combining model and data explanations involves composing these two functions to return a unified explanation as a set of provenance graphs. We denote such composition as XEE_h (eXplanations End to End relative to model h):

$$XEE_h(\mathbf{x}) = \{prov(d_i) | d_i \in \mathcal{I}(h(\mathbf{x}))\}$$

The main contribution of this paper is a demonstration of how such composition may be realised in a practical setting and a discussion of how the idea can be extended to more general settings. Specifically: (i) we have adapted the Tracin algorithm [4], available through the `Influenciae` python library https://github.com/deel-ai/influenciae and originally designed to operate on images and Keras-based models, to work on training sets that consist of tabular records; (ii) we use the PROLIT provenance generator [5] to track, retrieve, and describe the provenance of the influential training records; (iii) we demonstrate the connection between the two using the well-known Titanic dataset in combination with a simple data preparation pipeline and a binary classifier implemented as a simple feed-forward neural network.

The rest of the paper is organized as follows. In Section 2 we provide an overview of model explanations based on influence functions. In Section 3 we discuss data explanation describing PROLIT, a system we have developed for this purpose. In Section 4 we illustrate how model and data explanations can be combined and finally, in Section 5 we draw some conclusions and discuss future research on XEE.

2. Model explanations using influence functions

Research into methods that make "black box models" interpretable is not new, with notable model-agnostic algorithms like LIME (Local Interpretable Model-Agnostic Explanations) dating back from 2016 [6]. LIME is a local explanation technique that approximates the behavior of a complex model around a specific instance by locally fitting a simpler model around that instance.

It generates explanations by perturbing the input data and observing changes in predictions. The resulting surrogate model highlights the contribution of each feature to the prediction for that instance. LIME is particularly useful for understanding individual predictions, but it does not provide global insights into the behavior of the model [6].

Similarly, SHapley Additive exPlanations (SHAP) uses concepts from cooperative game theory to assign Shapley values to features, quantifying their contribution to a model's predictions. Unlike LIME, SHAP provides both local and global explanations. It is particularly effective for tree-based models and neural networks but can be computationally expensive for large datasets or highly complex models [7, 8].

The idea of using Influence Functions to support black-box explanations originates around the same time [9], but uses a distinctly different approach, aiming to trace a model's prediction through its learning algorithm and back to the training data. In practice, this entails answering the counterfactual question: how would the model's predictions change if a particular point was not included in the training set? This could be tested in theory by training the model of the entire training set, then selectively removing a single training point, retraining the model using the remainder of the set, and finally comparing the vectors of (optimised) weights in the two scenarios. The computational cost of this naive approach is prohibitive. However, it has been shown [9] that Influence Functions (IF) can be used to achieve practically good approximations of the actual relevance of each training data point while removing the need for retraining altogether. The key idea is that IF makes it possible to observe changes in the model's parameters as one single training point is "upweighted" by an infinitesimal amount. In practice, this amounts to "differentiating through the training" to estimate, in closed form, the effect of training perturbations.

A complete formalisation of the problem is beyond the scope of this description. Intuitively, the idea is to estimate the change in the parameters $\theta \in \Theta$ of the model due to removing a single point z from the training set. As we know, the training process calculates the optimal values for θ:

$$\hat{\theta} = arg\ min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

where $L(z, \theta)$ is the loss function and $z_1 \dots z_n$ are the training points.

The change in parameters can be expressed as $\hat{\theta}_{-z} - \hat{\theta}$, where

$$\hat{\theta}_{-z} = arg\ min_{\theta \in \Theta} \frac{1}{n} \sum_{z_i \neq z} L(z_i, \theta).$$

Influence functions remove the need to repeatedly retrain the model for each z in order to calculate $\hat{\theta}_{-z}$. The idea is to slightly *upweight* z in the loss function by a small perturbation ϵ, giving new parameters:

$$\hat{\theta}_{\epsilon,z} = arg\ min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta).$$

Results from influence functions theory show that this can be achieved by calculating the Hessian of $L(z, \theta)$, which does not require retraining but is itself computationally expensive. However,

the computational challenge of this calculation can be addressed using known approximation techniques, making the method usable in practice (see [9] for full details).

These findings paved the way for a number of variants of the methods, for instance [10, 11, 12], many of which are described in a recent survey [3]. For the purpose of our experiments we have used one of these, the Tracin algorithm [13], which is available as a Python library.[1]

3. Data Explanations using provenance

The provenance of tabular data that flows through a data science pipeline in preparation for use by machine learning algorithms is a graph representation of the sequence of transformations that alter both their schema and values. The PROLIT system [14] is designed to collect, store, manage, and query provenance information such as data flows through data preparation pipelines. It also leverages a Large Language Model (LLM) to allow users to query the provenance with questions expressed in natural language and to return short provenance *narratives* describing in words the operations performed on the data.

PROLIT relies on a provenance model that extends PROV [15] to capture provenance at different levels of granularity within a table, including atomic value, row, column, and whole table. This model, illustrated in Figure 1, represents provenance data with a graph structure involving Entity nodes (representing data values), Column nodes (representing entire columns of a dataset), and Activity nodes (representing operations on the data). Relationships between these nodes capture actions like data consumption (used), creation (GeneratedBy), removal (InvalidatedBy), and derivation (DerivedFrom) during data transformations. The belongsTo relationship links individual cells to their respective columns whereas the next relationship explicitly sequences activities.

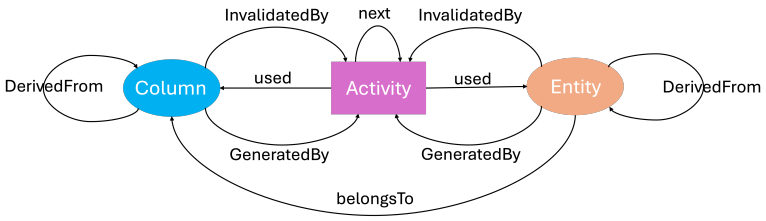

Figure 1: The data provenance model used in PROLIT

PROLIT relies on a LLM in two ways. Firstly, it recognises individual operations performed on the data from an arbitrary Pandas program and creates a structured representation of their transformations, rewriting user-defined pipelines into a standardised format. This code segmentation allows PROLIT to precisely associate provenance to each code snippet. Secondly, it generates human-readable narratives of both the code snippets and of elements of the provenance graph in response to a user question.

More in detail, PROLIT takes a Python script that manipulates Pandas dataframes as input. It first rewrites the script, using a LLM to identify and document the activities. The PROLIT

[1] https://deel-ai.github.io/influenciae/

provenance generator then observes the execution of the rewritten script, producing a provenance graph fragment that captures input/output data dependencies for every activity. The complete provenance graph at the end of execution is written into a graph database (Neo4j), where it can be analysed using either native Neo4j access or through a LLM-based component that translates natural language questions into Cypher queries and then renders the resulting graph as a natural language narrative.

An example of provenance captured by PROLIT for a pipeline that first eliminates from a dataset the Purchased column, which acts as the target of a classification model, and then imputes null values occurring in the Age column is reported in Figure 2.

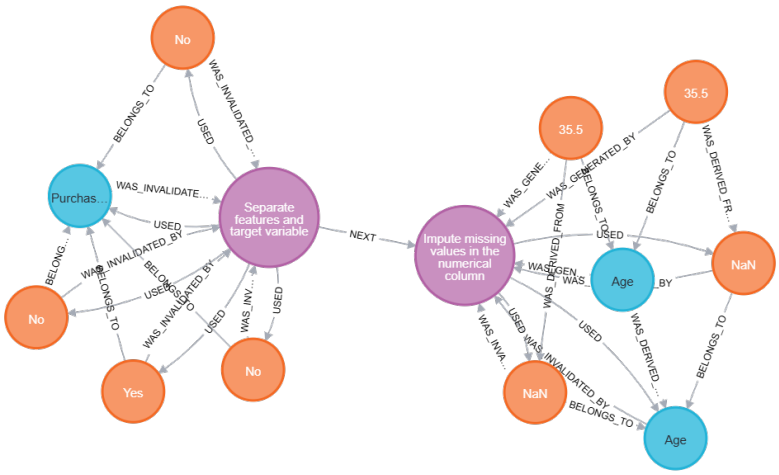

Figure 2: An example of data provenance collected by PROLIT. Orange nodes represent entities, blue nodes represent columns, and purple nodes represent activities.

PROLIT addresses the problem of the excessive volume of provenance data that can be generated when tracking individual elements of a large dataset by enabling customization of the level of granularity to which the provenance is collected and queried. In particular, it can provide a sketch of the collected provenance, involving a few instances for each operation in the pipeline, and a view of the provenance at schema-level, showing only operations and involved columns, hiding their effect on individual data.

4. XEE: a proof-of-concept end-to-end example

We now illustrate how model and data explanations can be combined, with a simple example based using the well-known Titanic tabular dataset, here denoted simply D. Firstly we describe D and the script S used to prepare D for model training. The model itself is a simple Multilayer Perceptron and is not shown in the detail. Then we perform a model prediction for a random test data point, and show the ranked list of influential records used in the prediction produced by `Tracin`. Finally, we query PROLIT with the top-ranking such record and display its provenance.

More in detail:

1. D is processed using script S. This results in training set $D' = S(D)$, derived from D through the operators in S;
2. PROLIT is used to collect the provenance graph $Prov^S_{D'}(d)$ of each element d in D' by observing the processing;
3. D' is split into training and test sets D'_{train}, D'_{test} and D'_{train} is used to train a simple neural network, producing a logistic regression classifier h where $\hat{y} = h(d) \in \{0, 1\}$ predicts whether or not a Titanic passengers survives.
4. The `Tracin` library, which implements the XAI approach described in [4], is used to generate explanations $\mathcal{I}(h(d)) = \{d_1 \dots d_n\}$, where $d_i \in D'_{train}$, for a specific instance $d \in D'_{test}$. Note that for the purpose of this exercise, we have adapted the reference Tracin implementation referenced above, which is designed to work on image data, to return records in a table as explanations.
5. Finally, PROLIT is used to retrieve the provenance $Prov^S_{D'}(d_i)$ for each $d_i \in \mathcal{I}(h(d))$. As discussed above, each of the provenance records is rendered using natural language narratives.

The Titanic dataset D comprises 12 attributes, shown in Fig.3.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

Figure 3: Titanic dataframe schema and attributes

The script S, reported in Figure 4, operates on D by performing three operations.

1. Drops unnecessary columns such as “Name”, “Ticket”, and “Cabin” which do not yield any useful information for the logistic regression model.
2. Imputes the mode for the “Embarked” column and the median for the “Age” column to handle missing values.
3. Performs one-hot encoding on categorical features “Pclass”, “Sex”, and “Embarked”.

This pipeline prepares the data to allow us to train our model, and as we are tracking provenance, we are able to investigate the data before it has been altered. `Tracin` runs alongside the training process as described in [4]. After training, when we run model inference on a random test data point (passenger ID 436), `Tracin` returns a ranked list of most influential training points (IDs 234, 775, and 262).

We should emphasise that validating influence, that is, understanding *why* these records have been ranked as highly influential, is an interesting question that is, however, beyond the scope of this example. In general, it would be interesting to study consistency across different influence-based methods, which are themeselves non-deterministic. In this example, one may speculate that the three survivors are “similar” to passenger 436 based on either their age, or sex.

We can now use PROLIT to retrieve the provenance of those passenger records, which will have been recorded when S is executed, focusing on one of the influential passengers (ID

```python
### Dropping irrelevant Columns
cols = ['Name', 'Ticket', 'Cabin']
df = df.drop(cols, axis = 1)

### Replace NAN in embarked with most frequent which is S
df['Embarked'] = df['Embarked'].fillna('S')
### Replace all NAN values with Median value
median_age_all = df['Age'].median()
df["Age"] = df['Age'].fillna(median_age_all)

### One Hot Encoding for categorical Features Pclass, Sex and Embarked
cols = ['Pclass', 'Sex', 'Embarked']
for i,col in enumerate(cols):
    dummies = pd.get_dummies(df[col])
    df_dummies = dummies.add_prefix(col + '_')
    df = df.join(df_dummies)
    df = df.drop([col], axis=1)
```

Figure 4: Pipeline S

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
436	1	1	Carter, Miss. Lucile Polk	female	14	1	2	113760	120	S
262	1	3	Asplund, Master. Edvin Rojj Felix	male	3	4	2	347077	31.38	S
775	1	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54	1	3	29105	23	S
234	1	3	Asplund, Miss. Lillian Gertrud	female	NaN	4	2	347077	31.38	S

Figure 5: Pandas dataframe with records from TraceIN results. Records are from D

234). PROLIT records the provenance of a record or single attribute, only when changes occur anywhere in the record. In this example, the state of the record for passenger 234 changes twice, firstly due to imputation, and then due to the one-hot encoding. These state changes are reflected in the provenance graph for this record, as shown in Fig.6. The top part of the figure shows the state of the record before and after each of the changes, while the bottom part shows the corresponding provenance derivations (read from the right "back" to the past on the left). Note that no other entities appear in the provenance record, because no other attribute values have changed.

5. Conclusions and future works

In this paper, we have argued for the importance of combining model-based explanations (XAI) with data explanations to create an eXplainable End-to-End (XEE) framework. By integrating techniques such as Influence Functions for model interpretability and a system that collects data provenance for pre-processing pipelines, we have demonstrated the feasibility of tracing predictions back to their raw data origins. This approach not only enhances the transparency

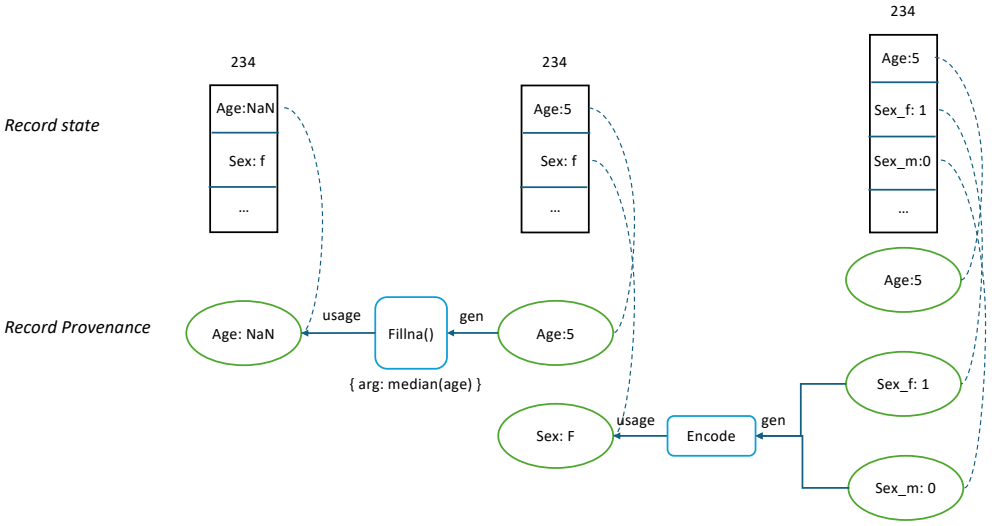

Figure 6: Provenance graph fragment for passenger 234

of machine learning models but also provides an improved understanding of the entire data-to-prediction pipeline, fostering a better explainability of AI systems.

We have presented a practical example of how "black box" explanations of model inference, obtained using Influence Functions, can be effectively combined with data explanations obtained by capturing and then querying the provenance of training data points through a pre-processing pipeline. The supporting example relies on the assumption that model explanations consist of training data points. This assumption is necessary so that the two explanations can be combined, however, it rules out alternative model explanation methods like SHAP, as this returns a ranking of attributes in order of relevance, *for the predicted data point itself*. This assumption is not too strict, given the demonstrated applicability of Influence Functions, however further insight into model explainability may be required to properly generalise the method and ensure that provenance can be safely composed with XAI methods, leading to full XEE.

There are indeed several possible directions of future research aimed at developing the XEE framework, including the following: (i) the current idea relies on Influence Functions and PROLIT, which may face scalability challenges with large datasets or complex models — future work could explore more efficient algorithms or approximations to make XEE applicable to real-world, large-scale systems; (ii) while Influence Functions are effective, they represent only one of many XAI methods — investigating how other XAI techniques, such as SHAP, LIME, or counterfactual explanations, can be integrated into the XEE framework would broaden its applicability and robustness; (iii) the effectiveness of XEE should be evaluated not only from a technical perspective but also in terms of its usability and interpretability by end-users, including domain experts and non-technical stakeholders — user studies could provide insights into how XEE explanations can be exploited in practice and tailored to different audiences; (iv) the current framework focuses on structured, tabular data — extending XEE to handle unstructured data, such as text, images, or time-series data, would expand its scope and utility.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] G. Schwalbe, B. Finzel, A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts, Data Mining and Knowledge Discovery 38 (2024) 3043–3101. URL: https://doi.org/10.1007/s10618-022-00867-8. doi:10.1007/s10618-022-00867-8.

[2] R. D. Cook, S. Weisberg, Characterizations of an empirical influence function for detecting influential cases in regression, Technometrics 22 (1980) 495–508. doi:10.1080/00401706.1980.10486199.

[3] Z. Hammoudeh, D. Lowd, Training data influence analysis and estimation: a survey, Machine Learning 113 (2024) 2351–2403. URL: https://doi.org/10.1007/s10994-023-06495-7. doi:10.1007/s10994-023-06495-7.

[4] G. Pruthi, F. Liu, S. Kale, M. Sundararajan, Estimating Training Data Influence by Tracing Gradient Descent, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 19920–19930. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/e6385d39ec9394f2f3a354d9d2b88eec-Abstract.html.

[5] A. Chapman, L. Lauro, P. Missier, R. Torlone, Supporting better insights of data science pipelines with fine-grained provenance, ACM Trans. Database Syst. 49 (2024). URL: https://doi.org/10.1145/3644385. doi:10.1145/3644385.

[6] M. T. Ribeiro, S. Singh, C. Guestrin, "Why Should I Trust You?" : Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16, ACM Press, New York, New York, USA, 2016, pp. 1135–1144. URL: http://dl.acm.org/citation.cfm?doid=2939672.2939778. doi:10.1145/2939672.2939778.

[7] A. Ghorbani, J. Zou, Data Shapley: Equitable Valuation of Data for Machine Learning, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 2242–2251. URL: https://proceedings.mlr.press/v97/ghorbani19c.html, iSSN: 2640-3498.

[8] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, C. J. Spanos, Towards efficient data valuation based on the shapley value, in: The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, 2019, pp. 1167–1176.

[9] P. W. Koh, P. Liang, Understanding black-box predictions via influence functions, in: International conference on machine learning, PMLR, 2017, pp. 1885–1894.

[10] H. Guo, N. F. Rajani, P. Hase, M. Bansal, C. Xiong, Fastif: Scalable influence functions for efficient model interpretation and debugging, 2021. URL: https://arxiv.org/abs/2012.15781. arXiv:2012.15781.

[11] J. Lin, A. Zhang, M. Lécuyer, J. Li, A. Panda, S. Sen, Measuring the Effect of Training Data on Deep Learning Predictions via Randomized Experiments, in: Proceedings of the

39th International Conference on Machine Learning, PMLR, 2022, pp. 13468–13504. URL: https://proceedings.mlr.press/v162/lin22h.html, iSSN: 2640-3498.

[12] D. Liu, P. Cheng, H. Zhu, X. Tang, Y. Chen, X. Wang, W. Pan, Z. Ming, X. He, Diwift: Discovering instance-wise influential features for tabular data, in: Proceedings of the ACM Web Conference 2023, WWW '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 1673–1682. URL: https://doi.org/10.1145/3543507.3583382. doi:10.1145/3543507.3583382.

[13] A. Picard, L. Hervier, T. Fel, D. Vigouroux, Influenciæ: A library for tracing the influence back to the data-points, 2024.

[14] P. L. Lazzaro, M. Lazzaro, P. Missier, R. Torlone, PROLIT: Supporting the Transparency of Data Preparation Pipelines through Narratives over Data Provenance, in: Proceedings of the 28th International Conference on Extending Database Technology (EDBT), OpenProceedings.org, Barcelona, Spain, 2025.

[15] P. Missier, K. Belhajjame, J. Cheney, The W3C PROV family of specifications for modelling provenance metadata, in: Procs. EDBT'13 (Tutorial), ACM, Genova, Italy, 2013. URL: http://www.edbt.org/Proceedings/2013-Genova/papers/edbt/a80-missier.pdf.