Chapter 3
The Right (Provenance) Hammer for the Job: A Comparison of Data Provenance Instrumentation

Adriane Chapman, Abhirami Sasikant, Giulia Simonelli, Paolo Missier, and Riccardo Torlone

3.1 Introduction

The W3C Provenance Working Group defines provenance as the "information about entities, activities and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness" (Mor 2013b,a). This statement by design gives no indication of how much information is needed to perform these assessments. Past examples have included as little as an original source (see ISO 19115-1:2014)[1] and as much as a full chain of processing for an individual data item (Brauer et al. 2014). While the collection and processing of provenance is important: to assess quality (Huynh et al. 2018), enable reproducibility (Thavasimani et al. 2019), reinforce trust in the end product (Batlajery et al. 2018), or to aid in problem diagnosis and process debugging (Herschel et al. 2017), what provenance is enough, and is it worth the cost of instrumentation?

[1]https://www.iso.org/obp/ui/#iso:std:iso:19115:-1:ed-1:v1:en

A. Chapman (✉) · A. Sasikant
University of Southampton, Southampton, UK
e-mail: adriane.chapman@soton.ac.uk; as1n16@soton.ac.uk

G. Simonelli · R. Torlone
Università Roma Tre, Rome, Italy
e-mail: giullia.simonelli@uniroma3.it; riccardo.torlone@uniroma3.it

P. Missier
Newcastle University, Newcastle upon Tyne, UK
e-mail: paolo.missier@ncl.ac.uk

L. F. Sikos et al. (eds.), *Provenance in Data Science*, Advanced Information and Knowledge Processing, https://doi.org/10.1007/978-3-030-67681-0_3

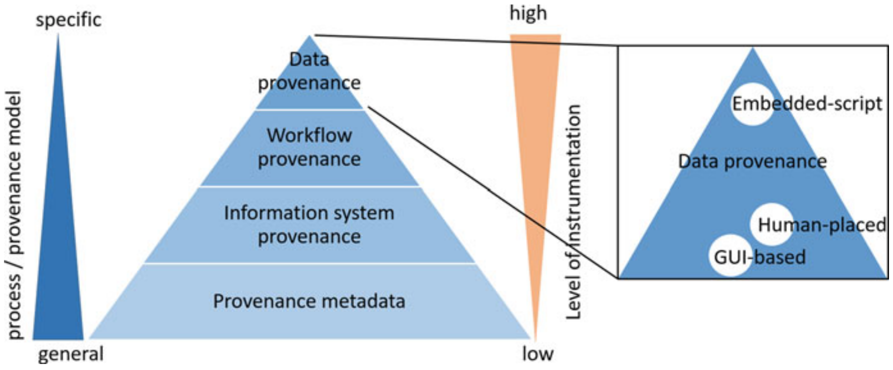


Fig. 3.1 Reproduction of Provenance Hierarchy from Herschel et al. (2017) and expanded to highlight the methods to instrument data provenance in the Orange3 framework in this work

Initial work and definitions of provenance (Buneman et al. 2001) evolved into many different types of provenance, including how, why, where (Cheney et al. 2009), and why not (Chapman and Jagadish 2009). Many surveys exist that create categorizations of different types of provenance, including provenance found in e-Science platforms (Simmhan et al. 2005), for computational processing (Freire et al. 2008), data provenance (Glavic and Dittrich 2007), and scripts (Pimentel et al. 2019). Based on a recent survey focused on looking at provenance and instrumentation needs (Herschel et al. 2017), provenance can be broadly classified into the following types: metadata, information system, workflow, and data provenance. In this context, Herschel et al. identified that with more specific information, such as data provenance, the level of instrumentation increases (Herschel et al. 2017).

Even within the “data provenance” category, there is a huge range in terms of instrumentation options available which can change the granularity of the provenance available for use. Within the hierarchy presented in Herschel et al. (2017), we investigate differences in instrumentation and supported queries of data provenance (Fig. 3.1).

In this work, we focus on a very specific set of tasks: collecting provenance of machine learning pipelines. There is a large amount of work involved in gathering and preparing data for use within a machine learning pipeline. Which data transformations are chosen can have a large impact on the resulting model (Feldman et al. 2015; Zelaya et al. 2019; Zelaya 2019). Provenance can help a user debug the pipeline or reason about final model results. In order to investigate granularity of data provenance and instrumentation costs, we use the application *Orange*[2] (Demšar et al. 2013) and show how provenance instrumentation choices can greatly affect the types of queries that can be posed over the data provenance captured. Our contributions include:

[2]https://orange.biolab.si

- We identified a tool, Orange3, that assists in organizing Python scripts into machine learning workflows, and use it to:
  - Identify a set of use cases that require provenance;
  - Constrain the environment so that a set of provenance instrumentation techniques can be compared.
- Implemented provenance instrumentation using a GUI-based insertion, embedded within scripts, and via expert hand-encoding in a machine learning pipeline.
- Compared the use of the resulting data provenance for each of these instrumentation approaches, in order to address issues the real-world use cases found in Orange3.

We begin in Sect. 3.2 with an overview of the Orange3 tool, and the provenance needs expressed by end users. In Sect. 3.3, we provide a brief overview of instrumentation options; Sects. 3.3.1, 3.3.2, and 3.3.3 discuss an instrumentation of ML pipeline scripts by instrumenting either hand-coded scripts (Sect. 3.3.1); the Orange3 GUI (Sect. 3.3.2); or by embedding directly in the scripts (Sect. 3.3.3). In Sect. 3.4, we compare these approaches with respect to solving the set of real-world problems identified in Sect. 3.2. We conclude in Sect. 3.5.

3.2 Case Study: Machine Learning Pipelines and Orange3

Machine learning is more than just the algorithms. It encompasses all of the data discovery, cleaning, and wrangling required to prepare a dataset for modeling and relies upon a person constructing a pipeline of transformations to ready the data for use in a model (Shang et al. 2019). In 1997, the development of Orange began at the University of Ljubljana. The goal was to address difficulties in illustrating aspects of machine learning pipelines with a standalone command-line utility. Since then, Orange has been in continuous development (Demšar et al. 2013), with the most recent release being Orange3.

Orange is an open-source data visualization, machine learning, and data mining toolkit. It provides an interactive visual programming interface for creating data analysis and machine learning workflows. It helps modelers gain insights from complex data sources. Orange can be utilized as a Python library, where Python scripts can be executed on the command line. The Orange package enables users to perform data manipulation, widget alterations, and add-on modeling. This platform caters to users of different levels and backgrounds, allowing machine learning to become a toolkit for any developer, as opposed to a specialized skill.

Orange features a canvas, a visual programming environment shown in Fig. 3.2, in which users create machine learning workflows by adding pipeline components called widgets. Widgets provide basic self-contained functionalities for all key stages of a machine learning pipeline. Each of these widgets is classified into categories based on their functionality and assigned priority, and is listed in the Orange toolbox. In addition, each widget has a description, and input/output

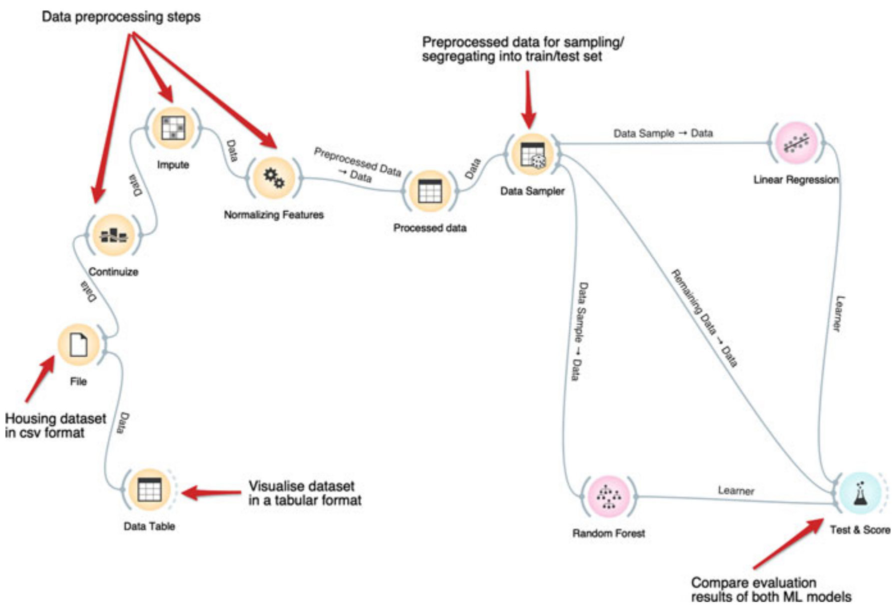


Fig. 3.2 Orange3 Canvas to support construction of machine learning pipelines. The pipeline shown is the “Housing” pipeline whose provenance is later shown in Figs. 3.3 and 3.7

channels associated with it. These widgets communicate and pass data objects, by means of communication channels. A data analysis workflow in Orange can be defined as a collection of widgets and communication channels.

The Orange library is a “hierarchically-organized toolbox of data mining components” (Demšar et al. 2013). The lower-level procedures are placed at the bottom of the hierarchy such as data processing and feature scoring, and higher-level procedures such as classification algorithms are placed at the top of hierarchy. The main branches of component hierarchy include data management and preprocessing, classification, regression, association, ensembles, clustering, evaluation, and projections. This library simplifies the creation of workflows and the generation of data mining approaches by enabling combinations of existing components. The key strength of Orange lies in connecting widgets in numerous ways by adopting various methodologies, resulting in new schemata (Demšar et al. 2013).

Ultimately, Orange allows users to organize and execute scripts that when placed together create machine learning pipelines.

3.2.1 Provenance Needs in Orange3

In general, provenance can be used to assess quality (Huynh et al. 2018), reproduce scientific endeavors (Thavasimani et al. 2019), facilitate trust (Batlajery et al. 2018), and help with debugging (Herschel et al. 2017). In this work, we focus on the debugging aspect. In order to gather real-world use cases of provenance in this domain, we reviewed a total of 370 use cases from the following forums: Data Science Stack Exchange (DSSE), Stack Overflow (SO), and the FAQ section on the Orange website (FAQ). From these 370 use cases, 12 use cases were considered relevant to the workflow debugging processes, and are listed in Table 3.1.

3.3 Overview of Instrumentation Possibilities

The Orange tool was created to facilitate creation of machine learning pipelines. While not implemented in Orange, there are some previous works on provenance in machine learning pipelines. Vamsa captures provenance of the APIs called and libraries used within a particular ML pipeline in order to help the user debug the pipeline (Namaki et al. 2020). However, the provenance captured does not focus on the data and what happened to it, instead on how the pipeline is constructed and the organization of scripts within it. Smaller than a pipeline, Lamp gathers provenance using a graph-based machine learning algorithm (GML) to reason over the importance of data items within the model (Ma et al. 2017). Finally, Jentzsch and N. Hochgeschwender recommend using provenance when designing ML models to improve transparency and explainability for end users (Jentzsch and Hochgeschwender 2019).

Because this work focuses on data provenance, we do not specifically review instrumentation options for script management (Pimentel et al. 2019), social interactions (Packer et al. 2019), or provenance from relational systems (Green et al. 2007, 2010). Provenance of scripts has recently been surveyed (Pimentel et al. 2019), with a classification of annotations, definition, deployment, and execution. In this work, the focus is effectively on *execution provenance*, "the origin of data and its derivation process during execution" of scripts.

Scientific workflows have been the earliest adopters of provenance in scripts, and we look to this community for inspiration in a machine learning pipeline creation setting. According to Sarikhani and Wendelborn (2018), provenance collection mechanisms have the ability to access distinct types of information in scientific workflow systems at the workflow level, activity level, and operating system level.

- Workflow-level provenance is captured at the level of scientific workflow systems. Here, the provenance capture mechanism is either attached to or is integrated within the scientific workflow system itself. A key advantage of this approach is that, the mechanism is closely coupled with the workflow system, and thus enables a direct capture process through the systems API. Within our machine learning pipeline creation context, this would be similar to capturing the design and configuration of the pipeline. We investigate this in Sect. 3.3.2.
- Activity(process)-level provenance captures provenance at the level of processes or activities of the scientific workflow. Here, the provenance mechanism is

Table 3.1 Uses for provenance identified in Stack Overflow (SO), Data Science Stack Exchange (DSSE)

QID	Source	Type	Description	Provenance needed
UC1	DSSE	Prediction	When applying the "Predictions" widget on the same training dataset, the results (i.e. probability scores) are different	The set of processes and their ordering in the execution.
UC2	DSSE	Prediction	When applying the "Impute" widget during preprocessing on train/test dataset, same values are predicted for all rows	The set of processes and their ordering in the execution.
UC3	DSSE	Classification/prediction	After performing image classification using an ML model, prediction probabilities are constant on test images	The processes and data of the execution; actual data changes required.
UC4	DSSE	Verification	From a constructed workflow using image classification (add-on widgets), ascertain whether the workflow performs "transfer learning"	The processes and data of the execution; actual data changes required.
UC5	SO	Prediction/Evaluation	Data fed to a handful of classifiers for comparing evaluation results fails to update, when using the "Rank" and "Test Learner" widgets in the workflow	The processes and data of the execution; the structure of the workflow.
UC6	DSSE	Rectification	Disproportionate allocation of labels after performing data analysis and modelling (inaccurate classification accuracy)	Fine-grained provenance showing detailed pipeline working to aid in logical understanding of widget utilization; actual data changes required.
UC7	DSSE	Prediction	Inaccuracy in the prediction of target variable using k-NN and linear regression ML models in an Orange workflow	The processes and data of the execution; actual data changes required
UC8	DSSE	Pipeline Construction	Ambiguity/difficulty in loading a data file (setting up the corpus) onto the Orange platform to perform data modelling using a linear regression model	Collective provenance of multiple workflows using the same ML model that would aid in predicting/suggesting possible widget combinations (paths)

UC9	DSSE	Prediction/Evaluation	Application of the “Test and Score” and “Predictions” widget on the same data utilizing the same ML model; produces differing results	The processes and data of the execution; actual data changes required
UC10	DSSE	Prediction	Differences in the predictions and corresponding goodness-of-fit R2 metric for the linear regression model on Orange and scikit-learn	The processes and data of the execution; actual data changes required
UC11	SO	Reproducibility of results	Differing results, clusters, and corresponding silhouette scores for the same processed dataset (including preprocessing steps); when using the “k-means” ML model widget	Widget connections and processes of the workflow; actual data changes are also required
UC12	SO	Evaluation	Deviations in the variance percentages accounted for by each of the principal components of the same dataset; for Principal Component Analysis (PCA) on Orange and scikit-learn	The processes and data of the execution; actual data changes required

independent from the scientific workflow system. In this approach, the mechanism requires relevant documentation, relating to information derived from autonomous processes, for each step of the workflow. In our context, this would be similar to capturing the information about the exact script that ran and what occurred to the data during that execution. This will be discussed in Sect. 3.3.3.
- Operating system-level provenance is captured in the operating system. This approach relies on the availability of a specific functionality at the OS level, and thus requires no modifications to existing scripts or applications. In comparison to workflow level, this mechanism captures provenance at a finer granularity. We omit this level here because the provenance at this level is unlikely to answer immediate user queries.

Looking beyond workflow systems, we highlight some of the classic architectural options (Allen et al. 2010) that could be used within the machine learning context and Orange.

Log Scraping
Log files contain much information that can be utilized as provenance. Collecting provenance information from these log files involves a log file parsing program. An example of this technique can be seen in Roper et al. (2020) in which the log files (change sets and version history files) are used to construct provenance information. Using log files means that developers keen on instrumenting provenance capture do not need to work within possibly proprietary or closed systems. On the other hand, there is little control as to the type and depth of information that can be obtained from log information. This approach could be used to gather either workflow-level or activity-level provenance from Sarikhani and Wendelborn (2018) depending on the set of logs available.

Human Supplied
The humans who are performing the work can sometimes be tasked to provide provenance information. Users of RTracker (Lerner et al. 2018) demonstrated that they were invested enough to carefully define and model the provenance needed in their domain. YesWorkflow (McPhillips et al. 2015; Zhang et al. 2017) allows users to embed notations within the comments of a code that can be interpreted by YesWorkflow to generate the provenance of scripts. These approaches effectively capture workflow-level provenance from Sarikhani and Wendelborn (2018). See Sect. 3.3.1 for human-supplied provenance in the Orange framework.

Application Instrumentation
A straightforward method for instrumenting provenance capture in any application is to modify the application directly. The major benefit of this approach is that the developer can be very precise regarding the information captured. The drawback is that the application must be open for developers to modify, and all subsequent development and maintenance must also take provenance into account. Applications that are provenance-aware cover the range from single applications, such as

provenance for visualization builder (Psallidas and Wu 2018), to larger workflow systems (Koop et al. 2008; Missier and Goble 2011; Santos et al. 2009; Souza et al. 2017). Other examples include provenance capture within MapReduce (Ikeda et al. 2012), Pig Latin (Amsterdamer et al. 2011), or Spark (Guedes et al. 2018; Interlandi et al. 2015; Psallidas and Wu 2018; Tang et al. 2019). This approach could be used to gather either workflow-level or activity-level provenance (Sarikhani and Wendelborn 2018). See Sect. 3.3.2 for application-based provenance capture in the Orange framework.

Script Embedding
Several approaches exist to embed directly into scripts. For example, NoWorkflow (Murta et al. 2015; Pimentel et al. 2017, 2016a) embeds directly into Python scripts and automatically logs provenance by program slicing. Other approaches use function-level information in Java to capture provenance information (Frew et al. 2008). In Sarikhani and Wendelborn (2018), this would be most appropriate for activity-level provenance capture. See Sect. 3.3.3 for script-embedding provenance capture in the Orange framework.

Recent work looks at different instrumentation vs. provenance-content choices available in systems that help scientists collect provenance of scripts (Pimentel et al. 2016a). NoWorkflow (Pimentel et al. 2017, 2016b) captures script execution by program slicing, while YesWorkflow (McPhillips et al. 2015; Zhang et al. 2017) asks users to hand-create capture at the desired places. The two systems are brought together in Pimentel et al. (2016a), and the types of provenance information have been compared. In the following sections, we look more closely at human supplied, application instrumentation and script embedding to gather provenance within the Orange toolkit.

3.3.1 Human-Supplied Capture

In order to understand what a human with basic provenance instrumentation tools at their disposal would do, we asked an expert provenance modeler to create a machine learning pipeline to predict median house values using linear regression and random forest on a housing dataset, and to instrument the code for provenance. A diagrammatic view of this workflow is shown in the Orange3 canvas in Fig. 3.2. This in effect reflects the baseline of what is currently obtainable by an invested human who has provenance modeling skills and is willing to take the time to insert calls to a provenance capture API, with carefully chosen information about processes, data, and agents. In this work, the expert used the standard *Prov Python*[3] libraries. Figure 3.3 shows the provenance generated by the expert for this pipeline.

[3]https://prov.readthedocs.io

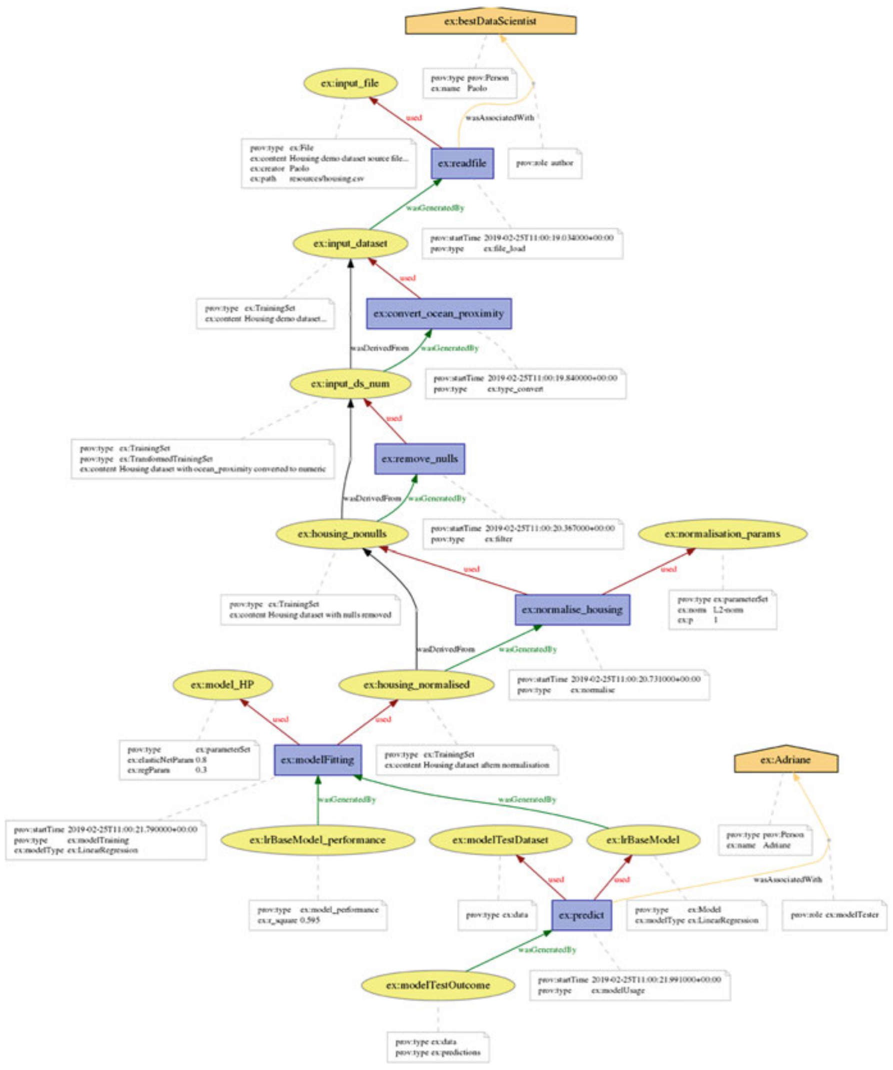


Fig. 3.3 Example of provenance captured via an expert provenance modeler for the machine learning pipeline

3.3.2 *GUI-Based Capture*

The provenance capture mechanism described in this section is built for Orange 3.16, released in October 2018. In order to capture provenance of the set of operations performed in Orange, while minimizing developer input in the creation and maintenance of provenance capture, we utilize the inherent class structure of the widgets used in the GUI.

Table 3.2 A subset of operations offered by Orange3 to build machine learning pipelines. The hook used by both the GUI and embedded approaches is noted

Orange3		GUI	Embedded
Type	Operator	Widget class	Transformation type
Data	File	OWFile	–
	SQL Table	OWSql	
	Select columns	OWSelectAttributes	Dimensionality reduction
	Select rows	OWSelectRows	
	Select relevant features	OWPreprocess	
	Select random features	OWPreprocess	
	Select data by index	OWSelectByDataIndex	
	Purge domain	OWPurgeDomain	
	Discretize	OWDiscretize	Feature transformation
	Continuize	OWContinuize	
	Randomize	OWRandomize	
	Impute	OWImpute	Imputation
	Edit domain	OWEditDomain	Value transformation
	Feature constructor	OWFeatureConstructor	Space transformation
	Create class	OWCreateClass	
Model	SVM	OWBaseLearner	–
	Linear regression		
	kNN		
	Tree		
	Stochastic gradient descent		
	Random forest		
	22 Other models		
Visualize	Box plot	OWBoxPlot	–
	Scatter plot	OWScatterPlot	

OWWidget, the parent widget class, is extended by all widget classes. This class provides all basic functionality of widgets regarding inputs, outputs, and methods that are fundamental to widget functioning. After analyzing the code associated with different widget categories, it was observed that this idea could be applied only to groups of widgets, with similar functionality such as model and visualization widgets; the parent classes extend OWWidget class for such widget groups. For example, as shown in Table 3.3, OWBaseLearner extends OWWidget and is the parent to all model widgets. Instrumenting OWBaseLearner for provenance capture provides provenance functionality to all modeling widgets.

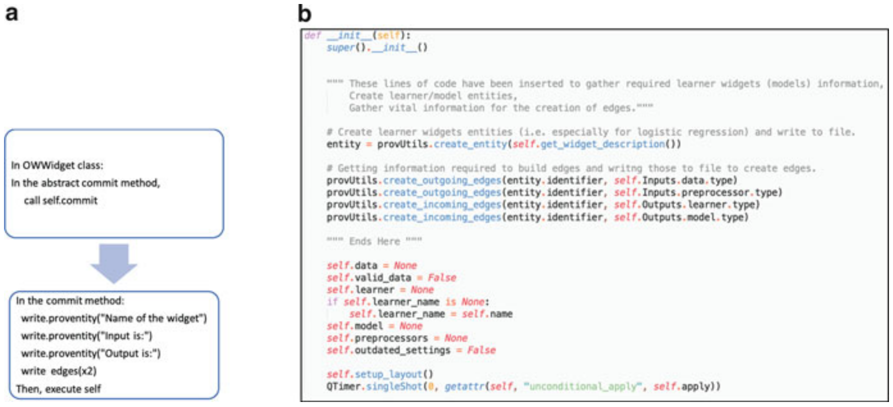
For other categories of widgets, it is necessary to capture provenance in each respective widget class. Because each of these widgets contains different input/output signal types and contents, the capture of this information is not standardized. Table 3.2 contains the set of Widget classes that are utilized to gather information for each operation. A diagrammatic representation of provenance cap-

Table 3.3 Parameters and provenance captured for each function type. (A) activities recorded; (E) entities recorded; (R) relationships recorded

Function type	Parameters	Provenance contains
Dimensionality reduction	out_dataframe description	A: a single activity, f, is created.
		R: the original entities are *invalidatedBy* f.
Feature transformation	out_dataframe columnsName description	A: a single activity, f, is created.
		E: a new entity is created for each modified item.
		R: the new entity *wasGeneratedBy* f; the new entity *wasDerivedFrom* the original entity; f *used* the set of feature items; the original entity *wasInvalidatedBy* f.
Space transformation	out_dataframe columnsName description	A: a single activity, f, is created.
		E: a new entity is created for every new attribute.
		R: f *uses* the set of entities that belong to the features used for the transformation; the new entities are *generatedBy* f and *derivedFrom* the entities of the related record and the features used for the transformation.
Instance generation	out_dataframe description	A: a single activity, f, is created.
		E: a new entity is created for every new instance.
		R: each new entity *wasGeneratedBy* f; f *uses* the existing entities that belong to the related feature.
Imputation (Dependent)	out_dataframe isIndependent=F description	A: a new activity for each feature is created.
		E: a new entity is created for every replaced value.
		R: each activity *uses* the entity for the related non-null feature; null entities are *invalidated*.
Imputation (Independent)	out_dataframe isIndependent=T description	A: a new activity for each feature is created.
		E: a new entity is created for every replaced value.
		R: each activity *uses* the entity for the related non-null feature; null entities are *invalidated*.
Value transformation	out_dataframe value description	A: a single activity, f, is created.
		E: a new entity is created for every replaced value.
		R: f *uses* the set of entities that belong to the features used for the transformation; the new entities are *generatedBy* f and *derivedFrom* the entities of the related record and the features used for the transformation.

ture involving OWWidget base class and code example of widget-based provenance capture are shown in Fig. 3.4. For more implementation details, please refer to Sasikant (2019).

In this section, we identify a capture point within the Orange3 architecture that allows provenance to be automatically captured as code instantiated by widgets gets executed. While this instrumentation is a "light touch" and can weather additional widgets that belong to predefined classes, it has no real insight as to what is happening to the data itself. By capturing at the GUI level, as the user drags and


a
In OWWidget class:
In the abstract commit method,
call self.commit
In the commit method:
write.proventity("Name of the widget")
write.proventity("Input is:")
write.proventity("Output is:")
write edges(x2)
Then, execute self
b

```python
def __init__(self):
    super().__init__()

    """ These lines of code have been inserted to gather required learner widgets (models) information,
        Create learner/model entities,
        Gather vital information for the creation of edges."""

    # Create learner widgets entities (i.e. especially for logistic regression) and write to file.
    entity = provUtils.create_entity(self.get_widget_description())

    # Getting information required to build edges and writng those to file to create edges.
    provUtils.create_outgoing_edges(entity.identifier, self.Inputs.data.type)
    provUtils.create_outgoing_edges(entity.identifier, self.Inputs.preprocessor.type)
    provUtils.create_incoming_edges(entity.identifier, self.Outputs.learner.type)
    provUtils.create_incoming_edges(entity.identifier, self.Outputs.model.type)

    """ Ends Here """

    self.data = None
    self.valid_data = False
    self.learner = None
    if self.learner_name is None:
        self.learner_name = self.name
    self.model = None
    self.preprocessors = None
    self.outdated_settings = False

    self.setup_layout()
    QTimer.singleShot(0, getattr(self, "unconditional_apply", self.apply))
```

Fig. 3.4 Provenance capture instrumentation via the Orange3 GUI. (**a**) Diagrammatic overview of GUI-based capture. (**b**) Example code required to add provenance capture to Orange widgets

drops operators, the provenance generated can be both retrospective and prospective (Lim et al. 2010).

3.3.3 *Embedded-Script Capture*

While the approach identified in Sect. 3.3.2 is a "light touch," it is not resilient with respect to Orange3 code changes and additions. It also does not provide the ability to introspect on changes to the data itself. In this approach, we instrument the Python scrips that execute the data preprocessing called upon execution of the pipeline.

Abstracting the functionality of the scripts that are executed by the Orange3 framework and represented by the Widgets in Orange3, there are several categories of functionality based on how the data itself is impacted or changed. These include dimensionality reduction, feature transformation, space transformation, instance generation, imputation, and value transformation, as shown in Table 3.2. Because script-embedded instrumentation is required to capture what happens with the data, and each type of operation does a different type of transform over the data, there are different provenance instrumentation calls for each type. However, the same type of instrumentation can then be reused for other scripts of the same category as shown in Table 3.3. Unfortunately, despite the abstract reuse of type of capture, every script must be individually provenance-instrumented. The architecture is shown in Fig. 3.5, and details of this implementation can be found in Simonelli (2019). Figure 3.6 contains a sample provenance record for a value transformation operation.

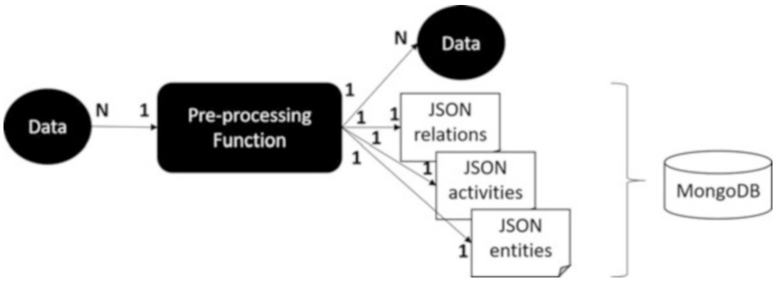


Fig. 3.5 The architecture for capture instrumentation for fine-grained provenance. Information in the machine learning pipeline is shown in black. Provenance artifacts are white

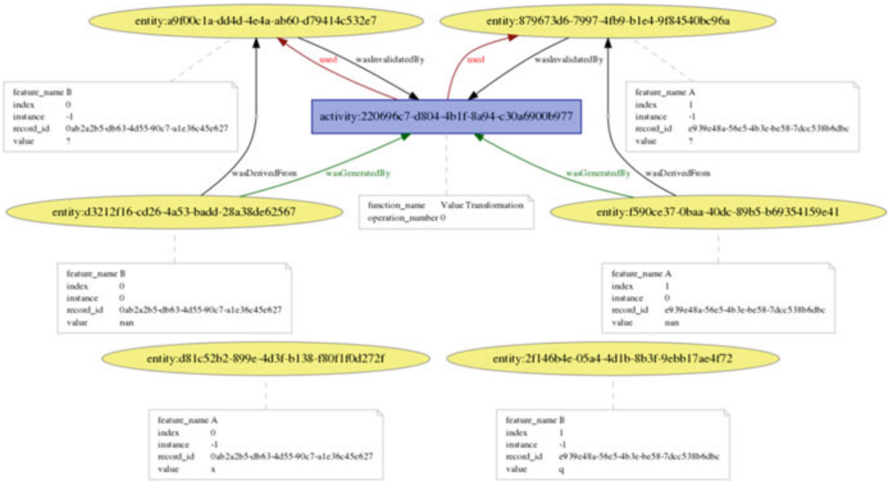

entity:a9f00c1a-dd4d-4e4a-ab60-d79414c532e7
entity:879673d6-7997-4fb9-b1e4-9f84540bc96a
used
wasInvalidatedBy
used
wasInvalidatedBy
feature_name B
index 0
instance -1
record_id 0ab2a2b5-db63-4d55-90c7-a1e36c45e627
value ?
activity:220696c7-d804-4b1f-8a94-c30a6900b977
feature_name A
index 1
instance -1
record_id e939e48a-56e5-4b3e-be58-7dcc538b6dbc
value ?
wasDerivedFrom
wasGeneratedBy
wasGeneratedBy
wasDerivedFrom
entity:d3212f16-cd26-4a53-badd-28a38de62567
function_name Value Transformation
operation_number 0
entity:f590ce37-0baa-40dc-89b5-b69354159e41
feature_name B
index 0
instance 0
record_id 0ab2a2b5-db63-4d55-90c7-a1e36c45e627
value nan
feature_name A
index 1
instance 0
record_id e939e48a-56e5-4b3e-be58-7dcc538b6dbc
value nan
entity:d81c52b2-899e-4d3f-b138-f80f1f0d272f
entity:2f146b4e-05a4-4d1b-8b3f-9ebb17ae4f72
feature_name A
index 0
instance -1
record_id 0ab2a2b5-db63-4d55-90c7-a1e36c45e627
value x
feature_name B
index 1
instance -1
record_id e939e48a-56e5-4b3e-be58-7dcc538b6dbc
value q

Fig. 3.6 Example of provenance captured for just one operation, value transformation, with a deeper instrumentation

3.4 Comparison of Instrumentation Approaches

We compare the data provenance and instrumentation requirements of our three approaches looking at provenance content (Sect. 3.4.1), the ability to answer real-world questions (Sect. 3.4.2), and the pros and cons of the instrumentation approach (Sect. 3.4.3).

3.4.1 Provenance Collected

We begin by comparing the three approaches with respect to the content of the provenance. The same machine learning pipeline that the expert provenance enabled

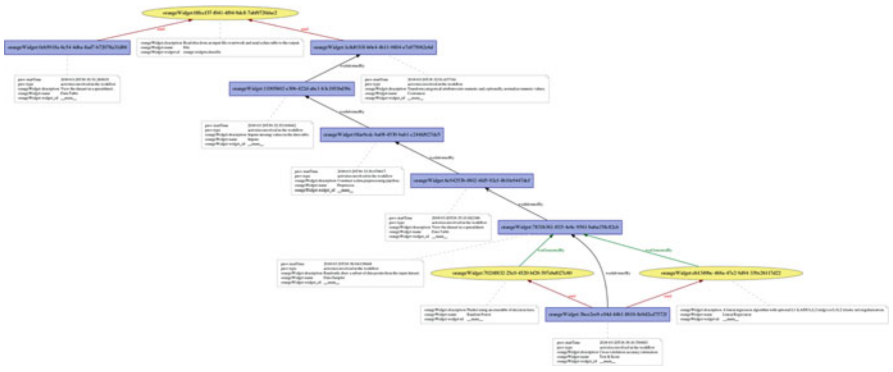


Fig. 3.7 Example of provenance captured via the GUI-based capture for the same machine learning pipeline that produced Fig. 3.3

was instantiated in Orange3, as shown in Fig. 3.2. The GUI-based provenance capture was applied, and produced the provenance shown in Fig. 3.7. On comparing the expert provided provenance graph (EP) in Fig. 3.3 with the GUI-generated provenance graph (GP) shown in Fig. 3.7 and the Script generated provenance (SP), the following can be seen:

- **Agents**: EP captured the presence of two agents. GP and SP do not have agents, as the Orange toolkit only involves one user at a time, who would be responsible for constructing the workflow.
- **Granularity of processes**: EP captured provenance at fine levels of granularity, providing minute details regarding processes involved and transformations of the specific dataset. For instance, the process of reading data from a file has been separated into three sub-stages: Entity [Input file] → Activity [Read file] → Entity [Input dataset]. Hence, provenance has been intricately captured in this scenario. On the other hand, GP and SP capture provenance at coarser granularity, providing key information of processes at a higher level. For the same process of reading data from file, provenance is captured widgetwise in GP, as widgets are building blocks of Orange workflows: Entity [File widget] → Activity [Data Table widget]. Hence, provenance of communication between widgets provides vital information (input/output signal components, widget details). However, GP does not capture provenance regarding data content changes, due to inaccessibility to the controller of Orange, and since provenance is captured at an abstract level, while SP captures large amounts of provenance about specific data changes.
- **Entity vs. activity**: Modeling decisions for entities and activities are different in EP, GP, and SP because of the organization imposed by the environment in which the capture calls were implemented.

In all approaches, provenance of all fundamental stages of the machine learning pipeline are captured: loading dataset, preprocessing/cleaning data, training models,

Table 3.4 Comparison of abilities to answer queries based on the information captured by a particular technique

ID	Human	GUI	Embedded
UC1	✓	✓	✓
UC2	✓	✓	✓
UC3	–	–	✓
UC4	–	–	✓
UC5	✓	✓	✓
UC6	–	–	✓
UC7	–	–	✓
UC8	✓	✓	✓
UC9	–	–	✓
UC10	–	–	✓
UC11	–	–	✓
UC12	–	–	✓

testing models on test data, and analysis of model predictions. There are some differences relating to detail in the provenance model, but for the most part, the three approaches capture the same type of information, with variations in detail and modeling.

3.4.2 Answering Provenance Queries

Returning to the Real-World use of provenance for machine learning pipeline development and debugging as described in Sect. 3.2.1, we now review which of these use cases can actually be resolved by the provenance captured via the three methods discussed in this work. Table 3.4 shows the Orange3 user queries that can be answered with each technique.

In essence, UC1, UC2, UC5, and UC8 contain questions that can be answered by understanding the overall pipeline design. They can be answered by understanding which data preprocessing functions were utilized, and the order in which preprocessing steps were made. The remainder of the use cases requires an analysis over the data itself, particularly spread and changes of spread in the data based on preprocessing.

3.4.3 The Cost of Provenance Instrumentation

Table 3.5 provides a high-level overview of the discussion within this section. There are two distinct roles that should be considered when contemplating data provenance instrumentation and usage: provenance modeler and provenance user. While in many cases, these are the same individual, the two roles require very different skills. A provenance modeler must:

Table 3.5 Comparison of costs for each method

	Human-placed	GUI-based	Embedded script
Developer/end user	Same	Can be different	Can be different
Num. files processed	Num. scripts written	$\leq$ Num. operators	Num. operators
Fragility of instrumentation	↓	↑	↓
Requires tool openness	↓	↑	↓
Constrains user to tool	↓	↑	↓
Size/detail of provenance	↓	↓	↑

- Understand how to model provenance. This includes the following:
  - Understand provenance, why it is used, and the important information required for capturing provenance including objects and relationships.
  - Understand the standards that are available.
  - Understand the application to be provenance enabled, and how the concepts in that application relate to provenance objects and relationships.
- Write the code to instrument the application.
- Maintain the provenance instrumentation throughout application updates.

The provenance user must:

- Interpret the provenance that is returned by the system to understand how it answers a given problem.

In many cases, particularly scientific exploration systems, these two roles are held by the same person. However, in large systems obtained through a formal acquisition process, e.g., for governments and large organizations, these two roles are filled by different people who may never interact with each other. In the context of machine learning pipelines and provenance instrumentation discussed in this work, human-supplied provenance requires that the roles of provenance modeler and provenance user are indeed held by the same person. Both the GUI-based and the Script-embedded allow the roles to be held by different individuals. This allows the end user of the provenance to be essentially provenance-unaware if they choose to be: they can merely be a consumer of data and not a developer.

However, there is a difference between GUI-based and script-based implementations. The GUI-based implementations, while providing less information about the data to the end user, are abstract enough that the provenance modeler comes up with fewer insertion points for capture calls within the 3rd-party code. In the worst case, GUI-based will insert as many capture calls as the number of operations, just as in the script-based implementations.

However, because we can utilize good code design, and occasionally embed the call in a parent class, it is sometimes possible for the GUIbased to have fewer calls than the script-based. This was done for OWBaseLearner, in which 1 file must be provenance enabled in the GUI-based method in order to capture provenance in 28 different learning models (see Table 3.2). Unless there is an upgrade to the GUI itself, any number of scripts and processing functions can be added to the underlying Orange3 tool, and the provenance modeler does not need to be aware of them. The GUI-based design is more fragile with respect to Orange3 refactoring and code updates. Unlike the script-embedding approach in which the fundamental machine learning Python scripts are provenance enabled, and suffer very little churn, the front end has experienced many code refreshes. Indeed, during the writing of this, the Orange Framework was undergoing yet another refactoring that likely changes the GUI-based provenance capture completely.

More generally and not constrained within Orange, the three approaches differ based on whether a tool is open or not. A human can embed provenance capture within their own scripts, and most underlying scripts are open. However, aiming for an application capture, like the Orange3, GUI-based approach requires that tool to be open. In the case of Orange3, this is true, but it may not be for many other applications. In a similar analysis, both human-based and script embedding allow a user to choose their tools of choice, while application-embedded implementations require the user to work within that single tool. Finally, the size of the provenance is similar between human-generated (e.g., Fig. 3.3 and GUI-based Fig. 3.7) implementations. Where the human is limited by time and inclination, the application is often limited by available hooks to information. In contrast, script-embedded capture can see and record all of the details, resulting in a larger provenance record (e.g., a small subset in Fig. 3.5).

3.5 Conclusions

In this work, we focused on an analysis of the types of data provenance that can be captured via very different instrumentation approaches within the same "task." In such a task, a user builds a machine learning pipeline and attempts to debug it. We identified many real-world scenarios of this process, documented in software development forums. We restricted ourselves to a set of Python scripts for processing data for machine learning, and a GUI-based tool, Orange, for organizing them. We showed how choices of provenance capture can greatly affect the types of queries that can be posed over the data provenance captured. We implemented provenance instrumentation using a GUI-based insertion, embedded within Python scripts, and via careful hand-encoding, applied to building a machine learning pipeline. We highlighted the instrumentation approaches possible by analyzing those used in scientific workflows, and described the different implementation approaches used within our narrowed domain. We then compared the utility of the resulting data provenance for each of these instrumentation approaches, to answer the real-world use cases found in Orange3. The results of this work provide a comparative analysis for future developers to identify and choose an apt instrumentation approach for future efforts.

Acknowledgments This work was partially supported by EPSRC (EP/S028366/1).

References

Allen MD, Seligman L, Blaustein B, Chapman A (2010) Provenance capture and use: a practical guide. the MITRE Corporation. https://www.mitre.org/sites/default/files/publications/practical-provenance-guide-MP100128.pdf

Amsterdamer Y, Davidson SB, Deutch D, Milo T, Stoyanovich J, Tannen V (2011) Putting lipstick on Pig: enabling database-style workflow provenance. In: Proceedings of the VLDB endowment, pp 346–357. https://doi.org/10.14778/2095686.2095693

Batlajery BV, Weal M, Chapman A, Moreau L (2018) Belief propagation through provenance graphs. In: Belhajjame K, Gehani A, Alper P (eds) Provenance and annotation of data and processes. Springer, Cham, pp 145–157. https://doi.org/10.1007/978-3-319-98379-0_11

Brauer PC, Czerniak A, Hasselbring W (2014) Start smart and finish wise: the Kiel Marine Science provenance-aware data management approach. In: 6th USENIX Workshop on the Theory and Practice of Provenance. https://www.usenix.org/system/files/conference/tapp2014/tapp14_paper_brauer.pdf

Buneman P, Khanna S, Tan WC (2001) Why and where: a characterization of data provenance. In: den Bussche JV, Vianu V (eds) Database theory – ICDT 2001. Springer, Heidelberg, pp 316–330. https://doi.org/10.1007/3-540-44503-X_20

Chapman AP, Jagadish HV (2009) Why not? In: Proceedings of the 2009 ACM SIGMOD international conference on management of data. ACM, New York, pp 523–534. https://doi.org/10.1145/1559845.1559901

Cheney J, Chiticariu L, Tan WC (2009) Provenance in databases: why, how, and where. Found Trends Databases 1(4):379–474. https://doi.org/10.1561/1900000006

Demšar J, Curk T, Erjavec A, Črt Gorup, Hočevar T, Milutinovič M, Možina M, Polajnar M, Toplak M, Starič A, Štajdohar M, Umek L, Žagar L, Žbontar J, Žitnik M, Zupan B (2013) Orange: data mining toolbox in Python. J Mach Learn Res 14(35):2349–2353. http://jmlr.org/papers/v14/demsar13a.html

Feldman M, Friedler SA, Moeller J, Scheidegger C, Venkatasubramanian S (2015) Certifying and removing disparate impact. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 259–268. https://doi.org/10.1145/2783258.2783311

Freire J, Koop D, Santos E, Silva CT (2008) Provenance for computational tasks: a survey. Comput Sci Eng 10(3):11–21. https://doi.org/10.1109/MCSE.2008.79

Frew J, Metzger D, Slaughter P (2008) Automatic capture and reconstruction of computational provenance. Concurr Comput: Pract Exp 20(5):485–496. https://doi.org/10.1002/cpe.1247

Glavic B, Dittrich KR (2007) Data provenance: a categorization of existing approaches. In: 12. Fachtagung des GI-Fachbereichs “Datenbanken und Informationssysteme”, University of Zurich, Zurich, pp 227–241. https://doi.org/10.5167/uzh-24450

Green TJ, Karvounarakis G, Tannen V (2007) Provenance semirings. In: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. ACM, New York, pp 31–40. https://doi.org/10.1145/1265530.1265535

Green TJ, Karvounarakis G, Ives ZG, Tannen V (2010) Provenance in ORCHESTRA. IEEE Data Eng Bull 33(3):9–16. http://sites.computer.org/debull/A10sept/green.pdf

Guedes T, Silva V, Mattoso M, Bedo MVN, de Oliveira D (2018) A practical roadmap for provenance capture and data analysis in Spark-based scientific workflows. In: 2018 IEEE/ACM Workflows in Support of Large-Scale Science, IEEE, pp 31–41. https://doi.org/10.1109/WORKS.2018.00009

Herschel M, Diestelkämper R, Lahmar HB (2017) A survey on provenance: what for? what form? what from? VLDB J 26:881–906. https://doi.org/10.1007/s00778-017-0486-1

Huynh TD, Ebden M, Fischer J, Roberts S, Moreau L (2018) Provenance network analytics: an approach to data analytics using data provenance. Data Mining Knowl Discov 32:708–735. https://doi.org/10.1007/s10618-017-0549-3

Ikeda R, Cho J, Fang C, Salihoglu S, Torikai S, Widom J (2012) Provenance-based debugging and drill-down in data-oriented workflows. In: 28th international conference on data engineering, IEEE, Los Alamitos, CA, USA, pp 1–2. https://doi.org/10.1109/ICDE.2012.118

Interlandi M, Shah K, Tetali SD, Gulzar MA, Yoo S, Kim M, Millstein T, Condie T (2015) Titian: data provenance support in Spark. In: Proceedings of the 42nd international conference on very large data bases, pp 216–227. http://www.vldb.org/pvldb/vol9/p216-interlandi.pdf

Jentzsch SF, Hochgeschwender N (2019) Don't forget your roots! Using provenance data for transparent and explainable development of machine learning models. In: 34th IEEE/ACM international conference on automated software engineering workshop, IEEE, Los Alamitos, CA, USA, pp 37–40. https://doi.org/10.1109/ASEW.2019.00025

Koop D, Scheidegger CE, Callahan SP, Freire J, Silva CT (2008) VisComplete: automating suggestions for visualization pipelines. IEEE Trans Visual Comput Graph 14(6):1691–1698. https://doi.org/10.1109/TVCG.2008.174

Lerner BS, Boose E, Perez L (2018) Using introspection to collect provenance in R. Informatics 5(1). https://doi.org/10.3390/informatics5010012

Lim C, Lu S, Chebotko A, Fotouhi F (2010) Prospective and retrospective provenance collection in scientific workflow environments. In: 2010 IEEE international conference on services computing, IEEE, Los Alamitos, CA, USA, pp 449–456. https://doi.org/10.1109/SCC.2010.18

Ma S, Aafer Y, Xu Z, Lee WC, Zhai J, Liu Y, Zhang X (2017) LAMP: data provenance for graph-based machine learning algorithms through derivative computation. In: Proceedings of the 11th joint meeting on foundations of software engineering. ACM, New York, pp 786–797. https://doi.org/10.1145/3106237.3106291

McPhillips T, Song T, Kolisnik T, Aulenbach S, Belhajjame K, Bocinsky K, Cao Y, Chirigati F, Dey S, Freire J, Huntzinger D, Jones C, Koop D, Missier P, Schildhauer M, Schwalm C, Wei Y, Cheney J, Bieda M, Ludäscher B (2015) YesWorkflow: a user-oriented, language-independent tool for recovering workflow information from scripts. https://arxiv.org/pdf/1502.02403.pdf

Missier P, Goble C (2011) Workflows to open provenance graphs, round-trip. Fut Gener Comput Syst 27(6):812–819. https://doi.org/10.1016/j.future.2010.10.012

Mor (2013a) Constraints of the PROV data model. http://www.w3.org/TR/2013/REC-prov-constraints-20130430/

Mor (2013b) PROV-DM: the PROV data model. https://www.w3.org/TR/prov-dm/

Murta L, Braganholo V, Chirigati F, Koop D, Freire J (2015) noWorkflow: capturing and analyzing provenance of scripts. In: Ludäscher B, Plale B (eds) Provenance and annotation of data and processes. Springer, Cham, pp 71–83. https://doi.org/10.1007/978-3-319-16462-5_6

Namaki MH, Floratou A, Psallidas F, Krishnan S, Agrawal A, Wu Y (2020) Vamsa: tracking provenance in data science scripts. https://arxiv.org/pdf/2001.01861.pdf

Packer HS, Chapman A, Carr L (2019) GitHub2PROV: provenance for supporting software project management. In: 11th international workshop on theory and practice of provenance. https://www.usenix.org/system/files/tapp2019-paper-packer.pdf

Pimentel JF, Dey S, McPhillips T, Belhajjame K, Koop D, Murta L, Braganholo V, Ludäscher B (2016a) Yin & Yang: demonstrating complementary provenance from noWorkflow & YesWorkflow. In: Mattoso M, Glavic B (eds) Provenance and annotation of data and processes. Springer, Cham, pp 161–165. https://doi.org/10.1007/978-3-319-40593-3_13

Pimentel JF, Freire J, Murta L, Braganholo V (2016b) Fine-grained provenance collection over scripts through program slicing. In: Mattoso M, Glavic B (eds) Provenance and annotation of data and processes. Springer, Cham, pp 199–203. https://doi.org/10.1007/978-3-319-40593-3_21

Pimentel JF, Murta L, Braganholo V, Freire J (2017) noWorkflow: a tool for collecting, analyzing, and managing provenance from Python scripts. Proc VLDB Endowm 10(12):1841–1844. https://doi.org/10.14778/3137765.3137789

Pimentel JF, Freire J, Murta L, Braganholo V (2019) A survey on collecting, managing, and analyzing provenance from scripts. ACM Comput Surv 52(3). https://doi.org/10.1145/3311955

Psallidas F, Wu E (2018) Provenance for interactive visualizations. In: Proceedings of the workshop on human-in-the-loop data analytics. ACM, New York. https://doi.org/10.1145/3209900.3209904

Roper B, Chapman A, Martin D, Cavazzi S (2020) Mapping trusted paths to VGI. ProvenanceWeek 2020, virtual event, poster

Santos E, Koop D, Vo HT, Anderson EW, Freire J, Silva C (2009) Using workflow medleys to streamline exploratory tasks. In: Winslett M (ed) Scientific and statistical database management. Springer, Heidelberg, pp 292–301. https://doi.org/10.1007/978-3-642-02279-1_23

Sarikhani M, Wendelborn A (2018) Mechanisms for provenance collection in scientific workflow systems. Computing 100:439–472. https://doi.org/10.1007/s00607-017-0578-1

Sasikant A (2019) Provenance capture mechanism for Orange, a data mining and machine learning toolkit, to evaluate the effectiveness of provenance capture in machine learning. Thesis, University of Southampton, Southampton

Shang Z, Zgraggen E, Buratti B, Kossmann F, Eichmann P, Chung Y, Binnig C, Upfal E, Kraska T (2019) Democratizing data science through interactive curation of ML pipelines. In: Proceedings of the 2019 international conference on management of data. ACM, New York, pp 1171–1188. https://doi.org/10.1145/3299869.3319863

Simmhan YL, Plale B, Gannon D (2005) A survey of data provenance in e-Science. ACM SIGMOD Record 34(3):31–36. https://doi.org/10.1145/1084805.1084812

Simonelli G (2019) Capturing and querying fine-grained provenance of preprocessing pipelines in data science. Thesis, Università Roma Tre, Rome

Souza R, Silva V, Coutinho ALGA, Valduriez P, Mattoso M (2017) Data reduction in scientific workflows using provenance monitoring and user steering. Fut Gener Comput Syst 110:481–501. https://doi.org/10.1016/j.future.2017.11.028

Tang M, Shao S, Yang W, Liang Y, Yu Y, Saha B, Hyun D (2019) SAC: a system for Big Data lineage tracking. In: 35th international conference on data engineering, IEEE, pp 1–2. https://doi.org/10.1109/ICDE.2019.00215

Thavasimani P, Caa J, Missier P (2019) Why-diff: exploiting provenance to understand outcome differences from non-identical reproduced workflows. IEEE Access 7:34973–34990. https://doi.org/10.1109/ACCESS.2019.2903727

Zelaya CVG (2019) Towards explaining the effects of data preprocessing on machine learning. In: 35th international conference on data engineering, IEEE, pp 2086–2090. https://doi.org/10.1109/ICDE.2019.00245

Zelaya VG, Missier P, Prangle D (2019) Parametrised data sampling for fairness optimisation. Explainable AI for fairness, accountability & transparency workshop, Anchorage, AK

Zhang Q, Morris PJ, McPhillips T, Hanken J, Lowery DB, Ludäscher B, Macklin JA, Morris RA, Wieczorek J (2017) Using YesWorkflow hybrid queries to reveal data lineage from data curation activities. Biodivers Inf Sci Stand 1:e20380. https://doi.org/10.3897/tdwgproceedings.1.20380