big data and cognitive computing

MDPI

Article

Traceability for Trustworthy AI: A Review of Models and Tools

Marçal Mora-Cantallops, Salvador Sánchez-Alonso, Elena García-Barriocanal and Miguel-Angel Sicilia *

Computer Science Department, Universidad de Alcalá, 28801 Madrid, Spain; marcal.mora@uah.es (M.M.-C.); salvador.sanchez@uah.es (S.S.-A.); elena.garciab@uah.es (E.G.-B.)
* Correspondence: msicilia@uah.es

Abstract: Traceability is considered a key requirement for trustworthy artificial intelligence (AI), related to the need to maintain a complete account of the provenance of data, processes, and artifacts involved in the production of an AI model. Traceability in AI shares part of its scope with general purpose recommendations for provenance as W3C PROV, and it is also supported to different extents by specific tools used by practitioners as part of their efforts in making data analytic processes reproducible or repeatable. Here, we review relevant tools, practices, and data models for traceability in their connection to building AI models and systems. We also propose some minimal requirements to consider a model traceable according to the assessment list of the High-Level Expert Group on AI. Our review shows how, although a good number of reproducibility tools are available, a common approach is currently lacking, together with the need for shared semantics. Besides, we have detected that some tools have either not achieved full maturity, or are already falling into obsolescence or in a state of near abandonment by its developers, which might compromise the reproducibility of the research trusted to them.

Keywords: trustworthy AI; artificial intelligence; traceability; provenance; replicability; reproducibility; transparency

Citation: Mora-Cantallops, M.; Sánchez-Alonso, S.; García-Barriocanal, E.; Sicilia, M.-A. Traceability for Trustworthy AI: A Review of Models and Tools. *Big Data Cogn. Comput.* **2021**, *5*, 20. https://doi.org/10.3390/bdcc5020020

Academic Editors: Michele Melchiori and Min Chen

Received: 2 March 2021
Accepted: 28 April 2021
Published: 4 May 2021

1. Introduction

The High-Level Expert Group on AI (AI HLEG) recently released a document with guidelines to attain "trustworthy AI" [1], mentioning seven key requirements: (1) human agency and oversight, (2) technical robustness and safety, (3) privacy and data governance, (4) transparency, (5) diversity, non-discrimination, and fairness, (6) environmental and societal well-being, and (7) accountability. Here, we are concerned with *transparency*, explained in the same document in terms of three components: traceability, explainability, and communication. These components should be applicable to all elements of the AI system, namely the data, the system, and the business model. Of the three components mentioned, communication is mostly related to the interface of the AI system, while explainability is a requirement on the decision process, related to the possibility of understanding the model or its functioning. In this work, we will discuss traceability, which should be "documented to the best possible standard", according to AI HLEG document. More concretely, the assessment list for traceability in the mentioned guidelines includes:

- Methods used for designing and developing the algorithmic system: how the algorithm was trained, which input data was gathered and selected, and how this occurred.
- Methods used to test and validate the algorithmic system: information about the data used to test and validate.
- Outcomes: The outcomes of the algorithms or the subsequent decisions taken on the basis of those outcomes, as well as other potential decisions that would result from different cases (for example, for other subgroups of users).

There are different data models and proposals oriented to fully document data, procedures, and outcomes for AI systems. These proposals typically focus on the tasks, configurations, and pipelines involved in machine learning (ML) models. A few of them

enable some form of automated repetition of the construction of the artifacts, although it is not clear if those tools and models per se are enough for the purposes of traceability for building transparent systems. For instance, Piccolo and Frampton [2] described seven tools and techniques for facilitating computational reproducibility, but noticed how none of those approaches were "sufficient for every scenario in isolation".

This can be illustrated with two examples. First, as described in [3], even using the same libraries, versions, and code for deep learning, models might lead to different results due to randomization and variability in implementing certain algorithms. On the other hand, the training of a classifier model might be automated with such tools so that it is technically repeatable, but lack descriptive elements that are critical for transparency [4], e.g., methods for obtaining the data, decisions taken in considering the model as suitable for a particular purpose, and how the outcomes are to be used once deployed, with a potential impact on users.

Improving the transparency, reproducibility, and efficiency of scientific research is key to increase the credibility of the published scientific literature and accelerate discovery [5]. However, some authors argue that machine learning, similar to many other disciplines, faces a reproducibility crisis [6]. In particular, the authors in [6] highlight how repeating and reproducing results and reusing pipelines is difficult, as "building an ML pipeline requires constant tweaks in the algorithms and models and parameter tuning", "training of the ML model is conducted through trial and error" and, as also mentioned in [3], "randomness in ML experiments is big" and its impact applies to many of the algorithmic steps, so it is not uncommon to have several runs of the model with the same data generating different results. They also point out how provenance information is key for reproducibility. Finally, ref. [7] noted that designing technology that supports reproducible research also implied a "more guided and efficient research process through preservation templates that closely map research workflows".

In an attempt to contribute to advancing practice from automation to comprehensive traceability, we review in this paper the models and tools oriented to document AI systems under the light of the AI HLEG guidelines, contributing to the field by providing an overview of the strengths and weaknesses of such models and tools. Then, we propose the elements of a minimal description profile of the metadata needed to enhance model traceability, combining descriptions and semantics that have been already used in other contexts.

Some of the problems to address can be exemplified if we think of the need for traceability. In this domain, traceability describing the artifacts, activities and actors involved in the production of the model or analysis is of course essential, but it also is for the description of the intentions, use case or rationale for selecting business cases. As for the description itself, it could be represented using standards as the W3C PROV model—in fact a proposal to extend the W3C PROV model for machine learning called PROV-ML [8] is available. However, these models are just means of expression, so further requirements are needed to use them in particular ways.

Traceability intersects with the concepts of *reproducibility* and *replicability* of data analysis. There is some terminological confusion with the concepts of reproducibility and replicability that have been discussed by Plesser [9]. In particular, the Association for Computing Machinery (ACM) adopted the following definitions in 2016, which added an additional step in the form of *repeatability* [10]:

- Repeatability (Same team, same experimental setup): The measurement can be obtained with stated precision by the same team using the same measurement procedure and the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.
- Replicability (Different team, same experimental setup): The measurement can be obtained with stated precision by a different team using the same measurement procedure and the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments,

this means that an independent group can obtain the same result using the author's own artifacts.

- Reproducibility (Different team, different experimental setup): The measurement can be obtained with stated precision by a different team and a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts that they develop completely independently.

Repeatability, is thus something that should be expected of any system; a non-repeatable result, not even by the same team that produced it in the first place, are seldom appropriate for publication. On the other hand, while *reproducibility* could be considered as the final objective, the guidelines for trustworthy AI by the AI HLEG are firstly concerned about assuring its intermediate step, *replicability*, allowing different individuals or teams to *replicate* a well-documented experiment obtaining the same (or similar) result using the same data.

Goodman et al. [11] further refine the previous definitions and try to solve the terminology confusion introducing a different wording:

- Methods reproducibility: provide sufficient detail about procedures and data so that the same procedures could be exactly repeated.
- Results reproducibility: obtain the same results from an independent study with procedures as closely matched to the original study as possible.
- Inferential reproducibility: draw the same conclusions from either an independent replication of a study or a reanalysis of the original study.

Here, a few things must be noted. Methods reproducibility is, thus, equivalent to the idea of replicability in the ACM definition, and it will be the focus of this review, as it is also the requirement of trustworthiness of any study that is mentioned in the work of the AI HLEG. On the other hand, results reproducibility would be closely related to the definition of reproducibility by the ACM. Finally, it is worth mentioning that inferential reproducibility would go even further in trustworthiness and represent an ultimate (and ideal) goal for trustworthy research altogether. Beyond terminological discussions, here we are mostly concerned with the practice of traceability. This is why our approach departs from a review of existing relevant data models, and examines the support for traceability in software tools. Finally, we propose the essential elements that should be included in a traceable account of an AI model that could be used as a profile for devising tools that provide comprehensive aaccounts of traceability.

The rest of this paper is structured as follows. Section 2 will review the current existing data models that aim to describe models and provide traceability to AI experiments. Section 3 will run a similar review on the tools whose objective is to assist in capturing the environments, data, and decisions taken to be able to reuse, share, and reproduce the process as a whole. In Section 4, a proposal on a minimal description profile to ensure methods reproducibility (or replicability in terms of the ACM) will be provided. Finally, conclusions and future outlook will close the article.

2. Existing Data Models

Provenance, as defined by the PROV W3C recommendations, is "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". In our field, provenance data can be used to manage, track, and share machine learning models, but also to many other applications, such as to detect poisonous data and mitigate attacks [12].

Perhaps the first serious effort to define a provenance model for computationally intensive science experiments handling large volumes of data was that of Branco and Moreau [13], who defined an architecture for the creation of provenance-aware applications allowing for provenance recording, storage, reasoning, and querying. The proposed architecture, characterized by a strong emphasis on scalability, aimed to integrate provenance onto an existing legacy application by introducing a model consisting of four phases:

documenting, storing, reasoning, and querying. To document the execution of a process, i.e., creating documentation records, the authors used an existing and generic protocol for recording provenance called PReP [14], which defines a representation of process documentation suitable for service-oriented architectures. These documentation records, once created, are stored and indexed to facilitate their efficient location in a later reasoning phase, which takes place asynchronously. The final reasoning phase thus consists of analyzing the documentation records to extract data provenance in the form of metadata that will be made available to end users.

The PROV data model [15] is the main document of W3C recommendations. It defines a number of provenance concepts and includes a notation for expressing cases in the data model such as, for instance, the provenance of a given document published on the Web. Basically, PROV distinguishes core structures, forming the essence of provenance and divided into Types (e.g., Entity, Activity, or Agent) and Relations (such as Usage or Attribution), from extended structures catering for more specific uses of provenance. Extended structures are defined by a variety of mechanisms such as subtyping, expanded relations, new relations, and others. The data model is complemented by a set of constraints providing the necessary semantics to prevent the formation of descriptions that, although correct in terms of the types and relations of the model used, would not make sense. Making use of all these structures and restrictions, PROV allows for provenance descriptions as instances of a provenance structure, whether core or extended. Although not specific of AI experiments, the PROV model allows describing AI experiments as entities for which expressions stating their existence, use, the way they were generated, starting point, and other interrelations can be written.

A precursor of the PROV data model is the Open Provenance Model (OPM) [16], an easier to formalize, more lightweight model, in which the essential concepts of provenance such as entities, activities, and relationships are also present and thus can be used to model AI experiments. OPM represents the provenance of objects using a directed acyclic graph, enriched with annotations capturing further information related to execution. Provenance graphs in OPM include 3 types of nodes: Artifacts—an immutable piece of state—Processes—actions performed on or caused by artifacts—and Agents—contextual entities acting as a catalyst of a process. Nodes are linked to each other by edges representing causal dependencies. In this way, past processes—or even processes that are still running—are modeled, as OPM is aimed to explain how some artifacts were derived and never how future processes will work.

However, as Doerr and Theodoridou point out [17], generic provenance models such as the OPM or Provenir [18] present disadvantages due to their generic nature, e.g., do not describe the physical context of scientific measurement, and other issues.

OpenML [19] is an online platform for open science collaboration in machine learning, used to share datasets and results of machine learning experiments, integrated with some widely used libraries, such as Weka or Scikit-learn. OpenML allows for scientists to challenge the community with the publication of a dataset and the results that are expected to be obtained after analyzing it. Datasets are described in a separate page where all the information about them is accessible to collaborators: general description, attribution information, data characteristics, statistics of the data distribution and, for each task defined on the data, the results obtained. OpenML expects the challenger to express the data challenge in terms of task types: types of inputs given, outputs expected, and protocols that should be used. Other important elements in OpenML are “Flows”, i.e., implementations of a given algorithm for solving a given task, and “Runs”, i.e., applications of flows on a specific task. From the traceability perspective, the platform allows coordinating efforts on the same task: the progress made by all the researchers implied can be traced as part of the process of sharing ideas and results in the platform. Although providing integration with popular libraries and platforms such as Weka, scikit-learn, or MLR, unfortunately OpenML does not allow users to model the artifacts produced during experiments (and their lineage) in the same detail as other systems like Schelter et al. do [20].

ModelDB [21] is an open-source system to version machine learning models that allows us to index, track, and store modeling artifacts so that they may later be reproduced, shared, and analyzed. Users can thus record experiments, reproduce results, query for models and, in general, collaborate (a central repository of models is an essential part of ModelDB). As in other models, it also provides traceability with a big emphasis on experiment tracking: ModelDB clients can automatically track machine learning models in their native environments (e.g., Scikit-learn or Spark ML). In fact, the backend presents a common layer of abstractions to represent models and pipelines, while the front-end allows web-based visual representation and analysis of those models.

Schelter et al. [20] propose a lightweight system to extract, store, and manage metadata and provenance information of common artifacts in ML experiments. Through a straightforward architecture, both experimentation metadata and serialized artifacts are stored in a centralized document database. This information can be later consumed by applications—such as those running regression tests against historical results—or queried by final end users. The main aim is to overcome problems and issues derived from the non-standardized way that data scientists store and manage the results of training and tuning models, as well as the resulting data and artifacts (datasets, models, feature sets, predictions, etc.). This proposal introduces techniques to automatically extract metadata and provenance information from experiment artifacts and, more interestingly, defines a data model that allows storing declarative descriptions of ML experiments. It even provides the possibility to import experimentation data from repositories such as OpenML.

3. Practices and Tool Support

As reported by previous studies [22], most of the material published in data repositories does not guarantee repeatability or reproducibility. The main issue is found in capturing the software and system dependencies necessary for code execution, which even in those cases where the original researchers included some notes or instructions, the context or the workflow might be missing, effectively rendering the execution impossible or needing a significant amount of additional work.

Other common approaches for releasing data and/or code by researchers prove equally problematic. Depositing code and data on personal websites or in repositories such as GitLab or GitHub is often ineffective, as most of the time, neither the runtime environments nor the contextual and the system information are included [23]. Even supplemental data deposited on a journal's repositories is unreliable, as previous works have reported that the majority of such datasets are unavailable due to broken links [24].

Recently, multiple online tools have been released, which are mostly based on cloud storing and on the containerization technology Docker, that aim to provide the tools and means to capture the environments in which research is produced to be able to reuse, share and, finally, reproduce the process as a whole. The number of tools that completely or partially try to cover all these aspects is constantly growing and include, among others, the following projects:

- Code Ocean (https://codeocean.com/, accessed on 11 November 2020), "a cloud-based computational reproducibility platform" [25], which brings together leading tools, languages, and environments to give researchers an end-to-end workflow geared towards reproducibility, enabling its users to share and publish their code, data, workflows, and algorithms.
- Whole Tale (https://wholetale.org/, accessed on 11 November 2020), a free and open-source reproducibility platform that, by capturing data, code, and a complete software environment "aims to redefine the model via which computational and data-driven science is conducted, published, verified, and reproduced" [26].
- The Renku Project (https://datascience.ch/renku/, accessed on 11 November 2020), a combination of a web platform (Renkulab) and a command-line interface (Renku CLI) that combines many widely-used open-source tools to equip every project on the platform with resources that aid reproducibility, reusability, and collaboration.

- ZenML (https://zenml.io/, accessed on 11 November 2020) is an extensible open-source machine learning operations framework to create reproducible pipelines.
- Binder (https://mybinder.org/, accessed on 11 November 2020), an open source web service that lets users create sharable, interactive, reproducible environments in the cloud. It is powered by other core projects in the open source ecosystem and aims to create interactive versions of repositories that exist on sites like GitHub with minimal extra effort needed [27].
- Data Version Control (DVC) (https://dvc.org/, accessed on 11 November 2020) is an open source version control system aimed at machine learning projects and their models.
- Apache Taverna (https://taverna.incubator.apache.org/, accessed on 11 November 2020), an open source domain independent Workflow Management System (a suite of tools used to design and execute scientific workflows).
- Kepler (https://kepler-project.org/, accessed on 11 November 2020), designed to help scientists, analysts, and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines
- VisTrails (https://www.vistrails.org, accessed on 11 November 2020), an open-source scientific workflow and provenance management system that provides support for simulations, data exploration, and visualization. A key distinguishing feature of VisTrails is its comprehensive provenance infrastructure that maintains detailed history information about the steps followed in the course of an exploratory task.
- Madagascar (http://www.ahay.org/wiki/Main_Page, accessed on 11 November 2020), an open-source software package for multidimensional data analysis and reproducible computational experiments.
- Sumatra (http://neuralensemble.org/sumatra/, accessed on 11 November 2020), a tool for managing and tracking projects based on numerical simulation or analysis, with the aim of supporting reproducible research.

In spite of the long list, and although all tools claim to allow for "reproducible" research, the analysis on their outlined features (either in their websites or promotional materials, see Table 1) shows that most of them are far from being fully compliant with the assessment list for traceability in the AI HLEG guidelines. Methods reproducibility (or replicability) is not fully covered either.

Table 1. Comparison between tools that aim to support "methods reproducibility" research.

Tool	Environment	Code	Provenance	Data	Narrative	Alt. Outcomes	Integration
Code Ocean	Yes	Yes	No	Yes	No	No	Yes
Whole Tale	Yes	Yes	Yes	Yes	Yes	No	Yes
Renku	Yes	Yes	Yes	Yes	No	No	No
ZenML	No	Yes	No	Yes	No	No	No
Binder	Yes	Yes	No	No	No	No	No
DVC	Yes	Yes	No	Yes	No	No	No
Taverna	No	Yes	Yes	No	No	No	No
Kepler	No	Yes	Yes	No	No	No	No
VisTrails	No	Yes	Yes	No	No	No	No

First of all, not all tools serve the same purpose. Code Ocean, Whole Tale, and Renku opt for a more holistic approach, closer to the principles of replicability and, even though only Whole Tale pays attention to the narrative, they combine the computing environment, code, and data in a way that facilitates sharing and transparency, and in some cases they even allow for the integration of the resulting *capsules* or *tales* into published research articles. Narrative should be understood as not only the comments that usually document or describe the code structure, but also the textual elements that provide the reasoning behind the decisions taken by the researchers, that are used to discuss the results or that contain information about alternative workflows or limitations, among others, and it is a

critical descriptive element to be able to make sense of the data and workflow. Other tools seem to be more limited in scope. For instance, Binder focuses on providing a shareable environment where code can be executed, but does not cover the rest of the aspects, similar to what OpenML (with its online environment) and Madagascar (a shared research environment oriented to a few scientific disciplines) provide. The rest of the analyzed tools focus in essence on saving the workflow or in creating pipelines in order to be able to repeat the experiment, and storing the configuration, versions, and libraries used (Sumatra, for instance, aims to record such information automatically). Although this is, in any case, valuable information, a notable issue compared to other approaches is the lack of capability to reproduce the experiment under the same operating conditions and in the same location as the original, compromising not only replicability, but even repeatability too.

Overall, the more complete tools cover well the technical side of replicability, including environment, code, data, and provenance information. Narrative, however, seems to receive less attention; providing detailed information about the motivation of the researchers to gather and select a particular set of data and the reasoning behind the model construction and testing is critical for transparency and methods reproducibility. Additionally, no tool has been found that brings focus to the potential alternative decisions that would result from different cases, which are explicitly considered in the mentioned guidelines and could enrich the outcomes of replicable research. Finally, it is also worth noting that some of the analyzed tools are no longer supported or updated. For instance, VisTrails has not been maintained since 2016, and Sumatra's current version (0.7.0) dates from 2015; obsolescent or outdated tools compromise methods reproducibility as much as losing the procedures or the data.

4. Minimal Description Profile

Different requirements for traceability entail different levels of detail or granularity for provenance descriptions. We consider here that replicability of the process of obtaining the AI models (in the sense of being able to repeat the computational processes that led to the model) is a basic requirement, and tools described above can be used to maintain the information required for that software construction task. However, traceability goes beyond the computational steps in several dimensions, which we deal with in what follows.

We focus here on the processes, actors, and artifacts. However, there is a cross-cutting dimension that we are not dealing with here explicitly, which is that of recording the agents and actual events that arrange the pipelines or are the material authors or producers of the artifacts. The W3C PROV data model, as mentioned in Section 2) is a generic model that is appropriate for capturing that aspect, which is linking concrete agents to events in the temporal scale. So, it is expected that the descriptions for the different artifacts and processing steps would have associated PROV descriptions relating the `prov:Agents` (contributors or creators, in our case, typically data scientists or automated tools) to each of the instances of `prov:Entity` (any of the data, model, or report digital artifacts), along with the instances of `prov:Activity` that creates or transforms the entities (typically, steps in data processing pipelines).

The main traceability profile can be expressed in RDF triples. In Figure 1, fragments of an example description in Turtle syntax are provided for illustration of the main requirements and pitfalls. The namespaces referenced are in some cases existing ontologies, in others are hypothetical ones, since no clear candidate for the descriptive function sought have been found among existing ontologies. The description of even a simple pipeline would require a long description, so it is expected that a large part of these descriptions would be generated by tools and libraries themselves. Furthermore, this is exemplified here as a single RDF document, but in a tool it could be a set of documents spread at different files referencing each other.

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core> .
@prefix sw: <http://sw-portal.deri.org/ontologies/swportal#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix sc: <http://schema.org/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix pipe: <http://some-data-pipeline-ont.org> .
@prefix ml: <http://some-data-model-ont.org> .
@prefix soft: <http://some-library-ont.org> .

<#a-paper>
    a sw:IndividualPublication ;
    dc:title "Predicting Future Antibiotic Susceptibility using Regression-based
        Methods" .
    # ...
<#a-dataset>
    a sc:Dataset ;
    sc:distribution <#a-file> ;
    skos:related "SCTID:116154003";
    sc:variableMeasured <#a-variable> .
    #...
<#a-variable>
   a sosa:ObservableProperty;
   dc:title "age" ;
   skos:related "SCTID:424144002" .
   # ...
<#a-file>
    a sc:DataDownload ;
    sc:contentUrl "https://www.mass.gov/some-dataset/data.csv" .
<#a-pipeline>
   a pipe:Pipeline;
   pipe:step <#read-data> ;
   pipe:step <#model-selection> .
 # ...
<#read-data>
   a pipe:Step ;
   dc:title "Reading data" ;
   pipe:next <#model-selection> ;
   pipe:consumes <#a-file> .
<#model-selection>
   a pipe:Step ;
   pipe:applies <#random-search-process> .
<#random-search-process>
   a pipe:Process ;
   dc:title "Automated model selection" ;
   ml:applies ml:random-search ;
   ml:implementation <#a-model-sel> .
<#a-model-sel>
   a soft:Estimator ;
   soft:libraryUsed soft:scikit-learn-0.24.1 ;
   soft:module "sklearn.model_selection.RandomizedSearchCV" ;
   soft:param <#a-hiperparam-space> .
   # ..
<#param1>
    a soft:Param ;
    soft:paramname "estimator" ;
    soft:value "sklearn.linear_model.HuberRegressor";
    pipe:rationale "..." .
# ...
```

Figure 1. Fragments of RDF annotations of an example.

4.1. Describing Data

There are different situations for the use of data in AI models. The most basic situation is that of a model built from a pre-existing dataset (be it simple or complex in structure, small or large). In that case, we have a set of digital artifacts to be described. If we restrict ourselves to tabular data types (in the understanding of other types, such as images, would eventually be transformed to that form), we can use schemas describing observations. An example is the model described by Cox [28], which distinguishes between *observations* (defined as "an event or activity, the result of which is an estimate of the value of a property of the feature of interest, obtained using a specified procedure") and *sampling features* (defined as "a feature constructed to support the observation process, which may or may not have a persistent physical expression but would either not exist or be of little interest in the absence of an intention to make observations").

The base case for describing data is illustrated in the RDF fragments in Figure 1. The `#a-file` element describes the file and its location as a `distribution` of a `Dataset` (and a related publication). The dataset in turn should describe the variables and their context of acquisition. These may turn into complex descriptions of instruments, sampling methods, and other contextual elements. Those descriptions are addressed by existing observation and dataset models, but are not directly supported by tools, which take datasets as digital files and describe information referring to their physical format, which leaves the traceability of the digital artifacts to their sources to comments or references to external documents. This hampers the ability of tools to semantically interpret the data, leaving out possibilities for matching or looking for related data based on semantic descriptions. Semantic descriptions require the use of domain ontologies. In the example, references to the SNOMED-CT terminology can be found at instance and variable levels, using the vaguely defined `skos:related` property which should ideally be changed to others with stricter semantics.

Beyond the base case, there are some other cases that require separate attention, including the following. However, models like that of Cox prove to be sufficient to represent them, concretely:

- Models based on reinforcement learning. In this particular case, the typical scenario is that of a software agent actively gathering observations. The concept `ObservationContext` should then reference that specificity and the associated `ProcessUsed` (both in the same Cox model) should also refer to the algorithm itself.
- Models using active learning. In active learning, some form of user interface asks some expert for labels or some form of feedback on predictions. Again, `ObservationContext` and `ProcessUsed` are sufficient to refer to the fact that there is an algorithm (in this case different from the algorithm used to train on data) that has a particular criterion (e.g., some form of confidence in the predictions) as the procedure to elicit further observations.
- Models with incremental learning. In cases in which a model is updated by training or fine tuning on a previous model, there seems to not be any special requirements on describing data.

Traceability is also critical in model output, be that predictions or other output as association rules. Following the same modeling concepts, the predictions from a ML model may themselves be regarded as observations, and the procedure should be referring either to the model itself producing the outputs, or to some specification that is able to reproduce it, as in the case of an specification of a pipeline as described below.

Programming languages, libraries, or tools for data science in many cases allow the attachment of metadata to objects representing data. For example, it is possible to attach fields to a `DataFrame` object from the `pandas` Python library. However, these objects are typically transformed by functions or operations, and there is no built-in support to carry and automatically transform those metadata annotations. In the case of archived data, there are formats with built-in support for rich metadata and organizations. A notable cross-platform, mature example is the HDF5 format [29].

4.2. Describing the Processing Pipeline

If we take for granted the description of data, the subsequent problem is that of tracing back to that data across transformations. The concept of *pipeline* appears in some form in all the data science frameworks across technology stacks. In [30], there is an example on how metadata can be integrated in a particular framework to support traceability.

In a broad sense, a pipeline is an aggregate of transformations that can be described as a directed acyclic graph (DAG). In Figure 1, some basic structure of `Pipeline` and `Step` instances is used, illustrating how to express the sequence of steps and the relation of steps with artifacts and the application of concrete computational processes. The minimal requisite for full traceability is that of adding meta-information on the transformations applied at each step of the pipeline, in all the cases in which they do not have a single interpretation. For example, deterministic transformations as changes of scale or well known aggregations are unambiguous, however there is still a problem of semantics in denoting them. However, in other classes, simple naming is not sufficient. For example, if we want to document the use of a decision tree model, we could use some shared open format as the Predictive Model Markup Language (PMML) maintained by the Data Mining Group). The use of the `TreeModel` class provides for conveying the structure of the tree itself, and a number of its parameters. However, not every aspect of the model can be conveyed with it. Missing aspects include the following:

- There is not an exhaustive schema for hyperparameters. An example could be some stopping criteria, as maximum depth of the tree, or less commonly found criteria for the quality of the splits. It is difficult to keep a schema updated with all the variations of different algorithm implementations.
- The process of selecting a model is done either by automated model selection or by the judgement of the analyst or by a combination of both. In the case of automated model selection, the selection algorithm becomes a node in the DAG, but in other cases the consideration of the finally selected model and the rationale for that selection is missing.
- Some algorithm implementations are dependant to some extent to the precision or the platform. The only way of precisely re-constructing the same model is referring to the actual code artifacts used, e.g., the concrete library release used in each step.
- Elements related to model quality are also missing in the schema, this notably includes the use of cross-validation and any other initialization or bootstraping done by the algorithms for the purpose of attempting to reach at better models. These are often implicit in the concrete implementations, but may in some cases be relevant in the attainment of adequate results.

Some of the tools discussed in Section 3 cover to some extent the aspects above, but do so implicitly in some cases. For example, dependencies to concrete versions of libraries are implicit in the fact that some form of *virtual environment* that makes a copy of the libraries used for dependency management. Furthermore, hyperparameter use can be identified combining explicit parameters in the code and default parameter values in the libraries. In the example fragments in Figure 1, this is illustrated with an instance of `Estimator` that represents a concrete library module. Note that this is complementary to terminologies as PMML that describe models in a generic way, not referring to concrete implementations.

4.3. Describing the Criteria for Evaluating Decision Making

An analytic system is not limited to a number of models producing outputs, but it also encompasses how these outputs are used to drive decision making. Except for models that are merely informative, this entails that there is some form of decision function from model outputs to a business action. From a modeling perspective, there is a need to describe the activity in which the model is used. As an example from the healthcare domain, we can consider the concept of *screening*, which is defined in OpenEHR CKM [31] as a "health-related activity or test used to screen a patient for a health condition or assessment of health risks."

The minimal elements to be included in the description are the following:

- The action that is immediately triggered by the decision. For example, in [32], a result of a prediction with "high probability" triggers an alert.
- The threshold or criteria that triggers the action. If some form of confidence in the prediction is to be used, it must be precisely recorded. Otherwise, it is not possible to provide complete accountability for the decisions of the system.

It should be noted that here we refer only to *immediate* actions. Following the example, the alert may then be followed by an appointment for a laboratory test, which in turn will led to an examination of its results by a physician, and so on. However, these subsequent steps are outside of the specifics of the model, and enter into the responsibility of the Information System that contains the models as components. The AI HLEG recommendations also include the business model as an element that is to be considered with regards to transparency. Many models use some sort of *profit-driven* criteria for model selection or construction, e.g., using some profit criteria instead of precision criteria [33]. This business model orientation is also present in established process frameworks for data mining or data science [34]. This is an example of relevant information piece for traceability, but is related first to the decisions taken in the phases of model building or selection, so it should be traced as such.

In the above discussion, we have addressed the main elements required for traceability if we aim for AI systems that are fully *replicable* and also allow for comparison and contrast, which requires a degree of semantic interoperability. Table 2 summarizes the main elements that need to be addressed. The Table may serve as a guide for a minimal set of requirements for tools and frameworks.

Table 2. Summary of the elements of the minimum description profile.

Phase (Based on CRISP DM)	Elements Required for Replicability	Elements Required for Semantic Interoperability
Business understanding	Recording business-oriented variables, related to expected outcomes (e.g., profitability)	Mapping those variables to domain terminologies.
Data understanding	Sources of data, be them static or continuously updating	(i) Mapping of observations and observable properties. (ii) Mapping of other contextual data elements.
Data preparation	Data transformation pipelines.	Mapping of processes to terminologies of transformation algorithms.
Modeling	Data modeling pipelines, incl. complete declarative reference of hyperparameters.	Mapping of processes to terminologies of model-producing algorithms.
Evaluation	Data evaluation pipelines, incl. selection criteria if not explicit in hyperparameters (as in automatic model selection)	Mapping of processes to terminologies of model-evaluating algorithms.
Deployment	(i) Recording the traces from prediction pipelines to outcomes produced. (ii) Recording of the decision models used and related actions (e.g., alerts).	Mapping of actions to domain ontologies, if relevant.
Cross-cutting (not in CRISP-DM)	Trace of agents and events producing each of the artifacts.	Provenance model (e.g., PROV)

It should be noted that, in some cases, the pipelines for data transformation, modeling, and selection are chained together in single pipelines with model selection, so that there will be criteria for model selection that also affects alternative data transformation choices.

5. Conclusions and Outlook

Traceability is a key component for the aim of transparent AI systems. A comprehensive approach to traceability would require on one hand a repeatable execution of the

computational steps, but also to capture aspects as metadata that may not be explicit or evident in the digital artifacts.

A number of tools for the purpose of reproducibility are available with different capabilities and levels of maturity, but a common approach is currently lacking. Future research should address this to fill that gap and enable interoperability across tools for traceability. In particular, it has been observed that most of the approaches are analogous to a record of transactions (instead of being closer to a researcher log or diary) and, thus, lack the ability to include and highlight the researcher's judgement and process of thought in their decision making. WholeTale might be the exception with its focus on narrative, but, even there, room for improvement has been identified in the form of the alternate decisions that would result from different cases and that are explicitly included in the AI HLEG assessment list. Lastly, it has also been observed how, while such tools have not achieved full maturity yet, many of them are already falling into obsolescence, lack updates, or have been abandoned by their developers. This raises yet another concern, as such outdated tools might also compromise the reproducibility of the research trusted to them.

Regarding the metadata needed to provide complete traceability, the first step is that of describing the data used as input for the creation of the models. There are ontologies that are able to convey all the details of the data as observations, including the phenomena observed and the context of the observation. The only remaining problem is that of having shared semantics, but this is not a problem of the metadata used for the annotations, but of the maturity of the descriptions of phenomena and contexts in different domains. In addition to the data, the processes applied to transform the data, and train and evaluate the models need to be described. This can be done by using DAGs that model the steps in the pipeline of data processing, but each of the steps in turn requires description. In the case of the model creation or training steps, languages as PMML could be used, but they lack the level of detail for complete repeatibility. Finally, the models themselves are just components in decision making processes, and this requires a description of those. Those descriptions are critical to trace the final outcomes of the model that impact business processes or users to the models themselves.

The elements discussed in the paper show a number of directions in which there is a need to carry out further work towards completely traceable decisions made based on models. This level of detail requires both common semantics (as can be provided by community-curated ontologies), but also support for annotations in data science tools, which are currently limited. An outline of a description profile that addresses all the phases in the production of AI systems has been proposed as a guide for future work in providing tools with interoperable and fully traceable processes.

Author Contributions: Conceptualization, M.-A.S.; methodology, M.M.-C. and M.-A.S.; validation, E.G.-B.; formal analysis, M.M.-C., S.S.-A. and M.-A.S.; investigation, M.M.-C., S.S.-A. and M.-A.S.; resources, E.G.-B.; data curation, M.M.-C.; writing–original draft preparation, M.M.-C., S.S.-A. and M.-A.S.; writing–review and editing, E.G.-B.; supervision, E.G.-B. and M.-A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. EU Commission. Ethics Guidelines for Trustworthy AI. 2019. Available online: https://ec.europa.eu/futurium/en/ai-alliance-consultation (accessed on 10 November 2019).
2. Piccolo, S.R.; Frampton, M.B. Tools and techniques for computational reproducibility. *GigaScience* **2016**, *5*, 30. [CrossRef] [PubMed]

3. Alahmari, S.S.; Goldgof, D.B.; Mouton, P.R.; Hall, L.O. Challenges for the Repeatability of Deep Learning Models. *IEEE Access* **2020**, *8*, 211860–211868. [CrossRef]
4. Anderson, J.M.; Wright, B.; Rauh, S.; Tritz, D.; Horn, J.; Parker, I.; Bergeron, D.; Cook, S.; Vassar, M. Evaluation of indicators supporting reproducibility and transparency within cardiology literature. *Heart* **2021**, *107*, 120–126. [CrossRef] [PubMed]
5. Munafò, M.; Nosek, B.; Bishop, D.; Button, K.S.; Chambers, C.D.; du Sert, N.P.; Simonsohn, U.; Wagenmakers, E.-J.; Ware, J.J.; Ioannidis, J.P.A. A manifesto for reproducible science. *Nat. Hum. Behav.* **2017**, *1*, 0021. [CrossRef]
6. Samuel, S.; Löffler, F.; König-Ries, B. Machine learning pipelines: Provenance, reproducibility and FAIR data principles. *arXiv* **2020**, arXiv:2006.12117.
7. Feger, S.S.; Dallmeier-Tiessen, S.; Schmidt, A.; Wozniak, P.W. Designing for Reproducibility: A Qualitative Study of Challenges and Opportunities in High Energy Physics. In Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '19), Glasgow, UK, 4–9 May 2019.
8. Souza, R.; Azevedo, L.; Lourenço, V.; Soares, E.; Thiago, R.; Brandão, R.; Civitarese, D.; Brazil, E.; Moreno, M.; Valduriez, P. Provenance Data in the Machine Learning Lifecycle in Computational Science and Engineering. In Proceedings of the 2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), Denver, CO, USA, 17 November 2019; pp. 1–10.
9. Plesser, H.E. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Front. Neuroinform.* **2018**, *11*, 76. [CrossRef] [PubMed]
10. Association for Computing Machinery. Artifact Review and Badging. 2016. Available online: https://www.acm.org/publications/policies/artifact-review-badging (accessed on 2 November 2020).
11. Goodman, S.N.; Fanelli, D.; Ioannidis, J.P.A. What does research reproducibility mean? *Sci. Transl. Med.* **2016**, *8*, 341ps12. [CrossRef] [PubMed]
12. Baracaldo, N.; Chen, B.; Ludwig, H.; Safavi, J.A. Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec '17), Dallas, TX, USA, 3 November 2017; pp. 103–110.
13. Branco, M.; Moreau, L. Enabling provenance on large scale e-science applications. In *International Provenance and Annotation Workshop*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 55–63.
14. Groth, P.; Luck, M.; Moreau, L. A protocol for recording provenance in service-oriented grids. In Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS'04), Grenoble, France, 15–17 December 2004.
15. Belhajjame, K.; B'Far, R.; Cheney, J.; Coppens, S.; Cresswell, S.; Gil, Y.; Groth, P.; Klyne, G.; Lebo, T.; McCusker, J.; et al. Prov-dm: The Prov Data Model. W3C Recommendation, 2013. Available online: https://www.w3.org/TR/prov-dm/ (accessed on 3 February 2021).
16. Moreau, L.; Freire, J.; Futrelle, J.; McGrath, R.E.; Myers, J.; Paulson, P. The open provenance model: An overview. In *International Provenance and Annotation Workshop*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 323–326.
17. Doerr, M.; Theodoridou, M. CRMdig: A Generic Digital Provenance Model for Scientific Observation. *TaPP* **2011**, *11*, 20–21.
18. Sahoo, S.S.; Sheth, A.P. Provenir Ontology: Towards a Framework for Escience Provenance Management. 2009. Available online: https://corescholar.libraries.wright.edu/knoesis/80 (accessed on 3 February 2021).
19. Vanschoren, J.; Van Rijn, J.; Bischl, B.; Torgo, L. OpenML: Networked science in machine learning. *SIGKDD* **2014**, *15*, 49–60. [CrossRef]
20. Schelter, S.; Boese, J.H.; Kirschnick, J.; Klein, T.; Seufert, S. Automatically tracking metadata and provenance of machine learning experiments. In Proceedings of the Machine Learning Systems Workshop at NIPS, Long Beach, CA, USA, 8 December 2017.
21. Vartak, M.; Subramanyam, H.; Lee, W.; Viswanathan, S.; Husnoo, S.; Madden, S.; Zaharia, M. ModelDB: A System for Machine Learning Model Management. In *Workshop on Human-In-the-Loop Data Analytics at SIGMOD*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 14:1–14:3.
22. Collberg, C.; Proebsting, T.A. Repeatability in computer systems research. *Commun. ACM* **2016**, *59*, 62–69. [CrossRef]
23. Rowhani-Farid, A.; Barnett, A.G. Badges for sharing data and code at Biostatistics: An observational study. *F1000Research* **2018**, *7*, 90. [CrossRef] [PubMed]
24. Pimentel, J.F.; Murta, L.; Braganholo, V.; Freire, J. A large-scale study about quality and reproducibility of jupyter notebooks. In Proceedings of the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), Montreal, QC, Canada, 25–31 May 2019; pp. 507–517.
25. Clyburne-Sherin, A.; Fei, X.; Green, S.A. Computational Reproducibility via Containers in Psychology. *Meta-Psychology* **2019**, *3*. [CrossRef]
26. Brinckman, A.; Chard, K.; Gaffney, N.; Hategan, M.; Jones, M.B.; Kowalik, K.; Kulasekaran, S.; Ludäscher, B.; Mecum, B.D.; Nabrzyski, J.; et al. Computing environments for reproducibility: Capturing the "Whole Tale". *Future Gener. Comp. Syst.* **2019**, *94*, 854–867. [CrossRef]
27. Jupyter, P.; Bussonnier, M.; Forde, J.; Freeman, J. Binder 2.0-Reproducible, interactive, sharable environments for science at scale. In Proceedings of the 17th Python in Science Conference, Austin, TX, USA, 9–15 July 2018; Volume 113, p. 120.
28. Cox, S.J. Ontology for observations and sampling features, with alignments to existing models. *Semant. Web* **2017**, *8*, 453–470. [CrossRef]

29. Folk, M.; Heber, G.; Koziol, Q.; Pourmal, E.; Robinson, D. An overview of the HDF5 technology suite and its applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, Uppsala, Sweden, 21–25 March 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 36–47.
30. Sicilia, M.Á.; García-Barriocanal, E.; Sánchez-Alonso, S.; Mora-Cantallops, M.; Cuadrado, J.J. Ontologies for data science: On its application to data pipelines. In *Research Conference on Metadata and Semantics Research*; Springer: Cham, Switzerland, 2018; pp. 169–180.
31. Sebastian Garde, O. Clinical Knowledge Manager. Available online: https://ckm.openehr.org/ckm/ (accessed on 30 April 2021).
32. Ichikawa, D.; Saito, T.; Ujita, W.; Oyama, H. How can machine-learning methods assist ual screening for hyperuricemia? A healthcare machine-learning approach. *J. Biomed. Inform.* **2016**, *64*, 20–24. [CrossRef]
33. Höppner, S.; Stripling, E.; Baesens, B.; vanden Broucke, S.; Verdonck, T. Profit driven decision trees for churn prediction. *Eur. J. Oper. Res.* **2020**, *284*, 920–933. [CrossRef]
34. Martínez-Plumed, F.; Contreras-Ochando, L.; Ferri, C.; Orallo, J.H.; Kull, M.; Lachiche, N.; Ramírez Quintana, M.J.; Flach, P.A. CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE Trans. Knowl. Data Eng.* **2019**. [CrossRef]