MLSea: A Semantic Layer for Discoverable Machine Learning

Ioannis Dasoulas(✉), Duo Yang, and Anastasia Dimou

KU Leuven – Leuven.AI – Flanders Make@KULeuven, Leuven, Belgium
{ioannis.dasoulas,duo.yang,anastasia.dimou}@kuleuven.be

Abstract. With the Machine Learning (ML) field rapidly evolving, ML pipelines continuously grow in numbers, complexity and components. Online platforms (e.g., OpenML, Kaggle) aim to gather and disseminate ML experiments. However, available knowledge is fragmented with each platform representing *distinct* components of the ML process or *intersecting* components but in different ways. To address this problem, we leverage semantic web technologies to model and integrate ML datasets, experiments, software and scientific works into **MLSea**, a resource consisting of: (i) **MLSO**, an ontology that models ML datasets, pipelines and implementations; (ii) **MLST**, taxonomies with collections of ML knowledge formulated as controlled vocabularies; and (iii) **MLSea-KG**, an RDF graph containing ML datasets, pipelines, implementations and scientific works from diverse sources. MLSea paves the way for improving the search, explainability and reproducibility of ML pipelines.

Keywords: Machine Learning · Ontologies · Knowledge Graphs · Semantic Web · Big Data Management

Resource Type: Ontology, Taxonomies and Knowledge Graph.
MLSea: https://w3id.org/mlsea.
MLSO Ontology: https://w3id.org/mlso.
DOI: 10.5281/zenodo.10286868 | **License:** CC BY 4.0.
MLSea-KG: https://w3id.org/mlsea-kg.
DOI: 10.5281/zenodo.10287349 | **License:** CC BY 4.0.

1 Introduction

As the field of Machine Learning (ML) continues to evolve, ML pipelines have grown in complexity, incorporating numerous systems, algorithms, datasets and hyper-parameters. Therefore, creating ML pipelines necessitates extensive research and a significant number of experiments to assess potential configurations, leading to expensive testbeds in terms of available resources and time [69].

To facilitate the construction of ML pipelines, several *general-purpose* (e.g., OpenML [46,68], Papers with Code [50] or Kaggle [34]) and *type-specific* (e.g., Hugging Face [27] and TensorFlow [65] which focus on deep learning) platforms and repositories catalogue ML experiments. Yet these platforms not only

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
A. Meroño Peñuela et al. (Eds.): ESWC 2024, LNCS 14665, pp. 178–198, 2024.
https://doi.org/10.1007/978-3-031-60635-9_11

exhibit variations in their data representations, access interfaces and search functionalities, but they also cover diverse aspects of ML pipelines. Hence, obtaining a holistic view of the entire ML pipeline process can be challenging, further hindering the already time-consuming and resource-intensive nature of the task and heightening the raising concerns about a reproducibility crisis in AI [23,28].

The searchability of ML pipelines can be improved by semantically enhanced representations that capture the entire spectrum of the ML process and embrace their diversity. This way, the discovery of ML knowledge can be facilitated by complementing ML knowledge across platforms. Efficient ML pipelines can then be achieved by discovering relevant datasets, publications, parameters and design choices from past experiments, without browsing over multiple platforms.

While past works have tried to model the various aspects of the ML process [16,37,49,54], they do not represent all aspects of the ML pipelines. Every framework is designed to model distinct data sources or aspects of the ML process, resulting in a scarcity of datasets and services that combine publicly accessible ML knowledge from a variety of sources.

To tackle these challenges, this work studies the ML pipeline, dataset attributes and software characteristics available in online repositories (e.g., OpenML, Kaggle, Papers with Code), and integrates their ML experiment data and metadata, resulting in our resource, **MLSea**, which makes these contributions:

(i) The **Machine Learning Sailor Ontology (MLSO)** that reuses and extends state-of-the-art ontologies to describe ML workflows, configurations, experimental results, models, datasets, and software implementations.
(ii) The **Machine Learning Sailor Taxonomies (MLST)**: 8 Simple Knowledge Organization System (SKOS) [43] taxonomies of ML-related concepts (e.g., task types, evaluation measures) with a combined total of 4532 SKOS concepts.
(iii) The **Machine Learning Knowledge Graph (MLSea-KG)**, a declaratively constructed and regularly updated KG with more than 1.44 billion RDF triples of ML experiments, regarding datasets used in ML experiments, tasks, implementations and related hyper-parameters, experiment executions, their configuration settings and evaluation results, code notebooks and repositories, algorithms, publications, models, scientists, and practitioners.

This resource provides a large-scale KG built from diverse sources related to ML pipelines descriptions, datasets attributes, and software implementations, accompanied by novel ML concepts taxonomies. MLSea provides a resource for sharing ML knowledge, potentially supporting the ML community in experiment documentation and reproducibility. This resource can be leveraged by ML and semantic web researchers and practitioners to discover ML workflows and metadata from a large source of ML experiments, conduct analyses and draw conclusions over ML experiments and their results, support ML recommendation systems, and complement Automated ML (AutoML) software systems.

In the next sections, we discuss motivating examples leading to the development of this resource (Sect. 2), related work (Sect. 3), the MLSO and MLST (Sect. 4), the construction process and use cases of MLSea-KG (Sect. 5), the potential impact of our resource (Sect. 6), and future work (Sect. 7).

2 Motivating Examples

Currently, the acquisition and discovery of multifaceted ML information require users to browse different ML repositories and platforms, where data are fragmented and disconnected from each other. We provide two motivating examples:

Example 1. *Find a dataset & its relevant code notebooks & scientific papers.* A user may search for the image dataset "CIFAR-10"[1] in OpenML [46], view its statistics and ML tasks defined on it. Yet, conducting a sole search on OpenML would not uncover the ML code notebooks on Kaggle [34] for this dataset[2] which can assist the users in programming their own ML projects. More, the user would not find scientific papers that introduced or referenced this dataset, as the ones included in Papers with Code[3], accompanied with code repositories. Such papers would enable the users to understand the scientific questions that can be tested with the dataset and the details of the ML pipeline. To gather information on various aspects of the ML process related to a specific dataset, multiple searches across different online repositories are needed.

Example 2. *Find a pipeline & its scientific papers, algorithms & parameters.* A user may search for implementations of the "Bagging" ensemble learning method in Papers with Code[4] and discover related scientific papers along with algorithms they may leverage. However, there also exist implementations in OpenML using this method that additionally provide hyper-parameters they are using, their descriptions and values[5]. The hyper-parameters can help users understand the configurations they would have to make and their different options when implementing this kind of pipeline.

3 Related Work

Semantic web technologies are regularly used to support data mining and ML systems in different stages of their pipeline [6,56]. ML systems leverage large knowledge graphs (KGs) to enhance their performance [8] and provide explanations regarding the models' decisions [31], taking advantage of the semantics and relationships captured within the KGs. In addition, semantic web technologies

[1] https://www.openml.org/d/40927.
[2] https://www.kaggle.com/datasets/pankrzysiu/cifar10-python/code.
[3] https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.
[4] https://paperswithcode.com/paper/bagging-provides-assumption-free-stability.
[5] https://www.openml.org/f/2058.

are regularly deployed to model ML pipelines, their different stages and components [16,37,48,54], in an effort to describe and document the ML lifecycle and the rich metadata that are associated with it.

Knowledge Graphs. Researchers have used semantic representations to better track, describe, and encode ML pipelines. In **Bosch** [72], formally encoded ML knowledge and solutions are translated to executable scripts, for external systems to leverage. **SemML** [73] enables the reuse and generalisation of ML pipelines for condition monitoring relying on ontology templates for ML task negotiation and data and feature annotation. **Venkataramanan et al.** [69] propose a knowledge-infused recommender system that utilizes metadata of executed ML pipelines from open repositories to recommend pipelines based on the users' queries. **AssistML** [70] collects and pre-processes metadata of existing ML solutions, to recommend alternatives for ML implementations. Regarding the tracking of ML publications, **Linked Papers with Code** [18] is a KG that provides information about ML publications from Papers with Code [50] with related metadata such as their datasets, tasks, and evaluations. Similarly, **SWeMLS-KG** [14] is a KG that contains workflows and machine-actionable metadata of 470 papers regarding systems that combine semantic web resources and ML components. To track ML datasets, **KGLac** [25] captures metadata and semantics of datasets to construct a KG that interconnects relevant datasets.

Despite the progress in encoding ML knowledge, there is an absence of large and openly available resources with ML knowledge from multiple sources and stages of the ML pipeline due to the complexity and diversity of the various ML repositories. While domain-specific KGs are commonly used in other areas, e.g., scientific work organization [4], publication tracking [19], and healthcare [39,57], the ML field lacks large resources describing and inter-connecting ML workflows.

Ontologies. Several ontologies have been curated to model ML systems, experiments, and pipelines. **EXPO** [63], one of the earlier attempts to model scientific experiments, formalizes the generic concepts of experiment design, methodology, and results representation and **Exposé** [67] focuses on describing ML experiments and components. **OntoDM** [48] provides generic representations and descriptions of the data mining domain and **DMOP** [37] describes data mining tasks, data, algorithms, hypotheses, and workflows, in an effort to support decision-making during the data mining process. **MEX Vocabulary** [16] and **PROV-ML** [64] describe ML experiments, with a strong focus on data provenance between stages of the ML lifecycle. **Task Ontologies** [72] model executable KGs of ML pipelines, describing pipelines as series of data, methods, and tasks. **ML-Schema** [54] is a collaborative effort from the W3C Machine Learning Schema Community Group[6], developed to align existing ML ontologies and serve as the foundation for more specific ontologies and applications.

These ontologies provide a solid foundation for describing the fundamental components within the ML domain. However, they are not integrated with other domain-specific ontologies. For instance, they are not aligned with ontologies about dataset and data catalogs description (e.g., DCAT Vocabulary [40], daQ

[6] https://www.w3.org/community/ml-schema/.

Ontology [11]), software systems and characteristics (e.g., SWO [41], SDO [22]), and scientific works (e.g., FaBiO [51]). More, there are no taxonomies covering pivotal ML concepts, such as algorithms, task types, fields, evaluation measures, and dataset characteristics. Whenever such taxonomies do exist within the ontologies (e.g., MEX Vocabulary [16]), they are limited in scope, and embedded within the ontologies as class taxonomies. This lack of modularity makes existing ML taxonomies hard to manage and integrate with other works, while future updates and additions may require extensive changes to the ontology structure.

ML Platforms. **Kaggle** [34] is a data science competition platform and online community of data scientists and ML practitioners. It contains numerous ML datasets, along with statistical analyses and code used to conduct ML experiments with them. **Papers with Code** is a community-driven platform which encompasses ML-related research papers together with their code, datasets, methods, and evaluation tables. **OpenML** [46,68] is an open platform for sharing datasets, algorithms, and experiments. It contains datasets along with ML tasks and pipelines in which they were employed. Additionally, it provides statistical metadata for the dataset characteristics and pipeline performance, and experiment configurations. **Hugging Face** [27] and **TensorFlow** [65] are popular platforms for collaboration in ML applications, focusing on the sharing of ML datasets and development of ML models.

4 The MLS Ontology and Taxonomies

The **Machine Learning Sailor Ontology (MLSO)** and the **Machine Learning Sailor Taxonomies (MLST)** provide a flexible schema to represent ML pipelines, datasets, implementations, and experiments. They are developed based on an in-depth analysis of ML experiment-related data from prominent online repositories, such as OpenML [46], Kaggle [34], and Papers with Code [50]. MLSO extends and enriches existing ontologies, such as ML-Schema [54], DCAT [40], SDO [22], and FaBiO [51] to model the ML pipeline. MLST consist of SKOS [43] controlled taxonomies to organize ML concepts. MLSO and MLST are available at https://github.com/dtai-kg/mlso/ and published at https://w3id.org/mlso/. A summary of their namespaces is available at Table 1.

Table 1. MLS Modules

Module	Description	Namespace
MLSO	Machine Learning Sailor Ontology	http://w3id.org/mlso/
MLSO-DC	Dataset Characteristic Taxonomy	http://w3id.org/mlso/vocab/dataset_characteristic/
MLSO-FC	Feature Characteristic Taxonomy	http://w3id.org/mlso/vocab/feature_characteristic/
MLSO-EM	Evaluation Measure Taxonomy	http://w3id.org/mlso/vocab/evaluation_measure/
MLSO-EP	Estimation Procedure Taxonomy	http://w3id.org/mlso/vocab/estimation_procedure/
MLSO-LM	Learning Method Taxonomy	http://w3id.org/mlso/vocab/learning_method/
MLSO-ALGO	Algorithm Taxonomy	http://w3id.org/mlso/vocab/ml_algorithm/
MLSO-F	Machine Learning Field Taxonomy	http://w3id.org/mlso/vocab/ml_field/
MLSO-TT	Task Type Taxonomy	http://w3id.org/mlso/vocab/ml_task_type/

4.1 MLS Development Methodology and Maintenance

The development of MLSO and MLST follows the Linked Open Terms (LOT) methodology [52]. The four major stages of LOT are: **Requirements Specification**, **Implementation**, **Publication** and **Maintenance**. We defined the ontology *requirements* based on the scope of the ontology, i.e. the discovery of ML knowledge from diverse platforms. The ontology was conceptualized using the draw.io [13] diagram generation tool and *implemented* in OWL2 [42] using Protegé [45]. To evaluate the ontology, SPARQL [24] queries[7] and the OOPS! validator [53] are used and its documentation is generated with Widoco [21].

Availability and Maintenance. For at least a ten-year period, KU Leuven will *maintain* the ontology and taxonomies with new requirements, error corrections, and gathering of recommendations, using the GitHub issue tracker of the ontology repository[8], where everyone can contribute and raise issues. We anticipate that this collaborative approach will foster the creation of a community around the ontology, that will take part in the ontology's further growth and development. Since we plan to add new knowledge and data from more platforms to our system, such as Hugging Face [27], the ontology will need to be carefully revised as newer versions will be published.

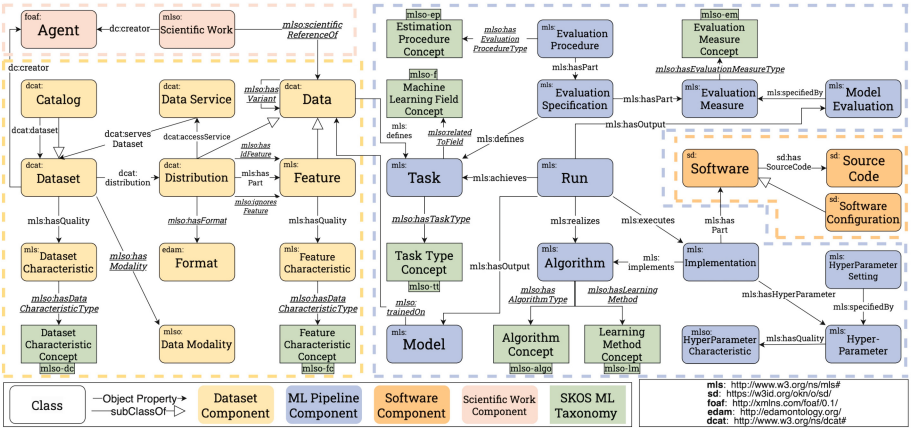

Fig. 1. MLSO and MLST Overview (Color figure online)

4.2 MLSO: The Machine Learning Sailor Ontology

The ontology's elements are grouped into four components: the **Dataset components** representing ML datasets along with their statistical and generic metadata; the **ML Pipeline components** representing the fundamental concepts of

[7] https://github.com/dtai-kg/MLSea-KGC/tree/main/sparql_examples.
[8] https://github.com/dtai-kg/MLSO.

the ML pipeline, such as the ML task and experiment; the **Software components**, representing software characteristics of the described implementations; and the **Scientific Work components**, representing ML peer-reviewed publications. The primary focus of the ontology lies in interconnecting these components to create a cohesive framework for modeling ML processes, extending already established standards. MLSO's core elements are shown in Fig. 1.

Dataset Components (yellow in Fig. 1) represent the concepts inherent to datasets utilized in ML processes. They describe the structural composition of datasets, along with other traits such as generic and statistical metadata. These concepts include: datasets, catalogs, data services, dataset distributions, formats, modalities, characteristics, dataset features, and feature characteristics.

To articulate the generic traits of dataset entities, such as their language, creators, and licences, we reuse the following general-purpose ontologies: the Data Catalog (DCAT), Friend Of A Friend (FOAF) [7], Provenance Ontology (PROV-O) [38], and Dublin Core (DC) [71]. While these ontologies capture the generic aspects of datasets, they do not provide statistical metadata crucial for ML processes or for defining ML tasks relevant to the datasets. To address this limitation we align DCAT and ML-Schema, and use ML-Schema to describe data qualities of datasets and dataset features, as well as ML tasks based on datasets. Moreover, for the description of dataset formats, we reuse the extensive data formats collection found in the EDAM Ontology [32].

While OpenML provides a means to represent data qualities[9], the existing representation is limited to a list of literals published on OpenML. To bridge this gap, we introduce 2 SKOS taxonomies: the Dataset Characteristic Taxonomy and the Feature Characteristic Taxonomy (Sect. 4.3). These taxonomies classify dataset and feature characteristics, respectively, providing a standardized approach to express the qualities of datasets and their features.

We introduce new properties to connect datasets (mlso:hasVariant), datasets with their features (mlso:hasDefaultTargetFeature and mlso:hasIdFeature) and their modality (mlso:hasModality), and data qualities with the introduced data quality types (mlso:hasDataCharacteristicType). We also introduce new properties to connect datasets and ML implementations that leverage them and models trained on these datasets (mlso:hasRelatedImplementation, mlso:trainedOn).

Last, the reused ontologies do not include a way to connect datasets with peer-reviewed publications that reference them or software that leverages them. Therefore, we added new properties to link datasets with relevant publications (mlso:hasScientificReference) and software using them (mlso:hasRelatedSoftware), such as code notebooks or large-scale software applications. We also include other dataset attributes, e.g., their data loader locations (mlso:hasDataLoaderLocation), as well as the cache format of their platform version (mlso:hasCacheFormat).

ML Pipeline Components (blue in Fig. 1) serve as a comprehensive representation of the fundamental concepts integral to the ML pipeline. They focus on describing the characteristics of ML experiments, their hyper-parameters, set-

[9] https://www.openml.org/search?type=measure.

tings, and evaluation results. The concepts include: ML tasks, runs, implementations, implementation characteristics, algorithms, hyper-parameters, hyper-parameter settings, experiments, models, model evaluations, model characteristics, evaluation measures, evaluation specifications, and evaluation procedures.

We reuse and extend ML-Schema to represent the fundamental concepts of ML pipelines, such as the tasks they address, their implementations and executions, and their inputs and outputs. While ML-Schema captures the fundamental concepts of the ML process, it does not include the details and characteristics of software used in these processes. To address this, we align ML-Schema and SDO, an ontology for describing software components, including their metadata and their inputs, outputs and variables. SDO enriches the representations of ML pipelines with their software characteristics and requirements.

ML-Schema offers descriptions of estimation procedures, evaluation measures, algorithms, and tasks. However, these are represented as literals without systematic classification or categorization. To more efficiently classify and document these concepts, we introduce 4 SKOS taxonomies; the Estimation Procedure, Evaluation Measure, Task Type, and Algorithm Taxonomies (Sect. 4.3), allowing for the classification of estimation procedures, evaluation measures, algorithms, and tasks, enriching the ontology with a categorized representation. More, we introduce 2 new taxonomies: the Machine Learning Field Taxonomy to classify ML fields of research that tasks belong to and the Learning Method Taxonomy to categorize learning methods used in ML pipelines (Sect. 4.3).

To explicitly assign types and categories to ML-Schema classes, we introduce new properties linking ML tasks to ML task categories (mlso:hasTaskType) and ML fields they relate to (mlso:relatedToField). We link ML algorithms with algorithm types (mlso:hasAlgorithmType) and types of ML learning method they may include (mlso:hasLearningMethodType). We also link evaluation measures and estimation procedures with their types (mlso:hasEvaluationProcedureType) and evaluation types (mlso:hasEvaluationMeasureType. Last, we introduce a new property (mlso:hasScientificReference) to link ML models or pipelines and the scientific works related to them or referencing them.

Software Components (orange in Fig. 1) encapsulate fundamental software concepts related to the execution of ML experiments, the settings and the requirements they have. These concepts include: software, software configuration, and software source code. We reuse SDO to describe the software characteristics and requirements of software used in ML pipelines. MLSO leverages these already established descriptions to augment the linkage between ML experiments and their software components. We introduce new properties to interconnect software components with scientific works (mlso:hasScientificReference), datasets (mlso:hasRelatedSoftware), and with ML fields of research (mlso:relatedToField).

Scientific Work Components (pink in Fig. 1) represent peer-reviewed publications related to developments in the field of ML. We reuse the FaBiO and FOAF vocabularies to describe publications, authors. MLSO extends FaBiO to connect ML publications with datasets they leverage or reference, ML experiments they conduct, models they create and publish, and the soft-

ware they use. We introduce new properties to enrich the ML research works descriptions, connecting them with datasets and software they leverage (mlso:hasScientificReference, mlso:hasRelatedSoftware), and ML fields of research (mlso:relatedToField).

4.3 MLST: Machine Learning Sailor Taxonomies

MLSO is complemented by MLST, a series of SKOS taxonomies, formulated as RDF vocabularies, to provide a flexible and standardized framework for managing large controlled collections of ML knowledge. These taxonomies serve as comprehensive thesauri, offering a structured overview of essential ML concepts.

The taxonomies were developed by systematically studying and integrating data from three primary sources. Firstly, data were gathered from online ML repositories that provide organised data collections, such as the OpenML data qualities collection[10] and the Papers with Code ML methods collection[11]. Secondly, information was sourced from smaller pre-existing collections formalized as vocabularies, such as the MEX vocabulary [16]. Lastly, categorizations were extracted from peer-reviewed publications that systematically consolidate ML knowledge, predominantly from ML surveys and scholarly works.

MLST include the following 8 ML-focused taxonomies:

(i) **Dataset Characteristic Taxonomy**, a collection of qualities used to characterize datasets (e.g., entropy, number of classes). It is based on the OpenML data qualities collection(see footnote 10) and contains 127 dataset characteristics.

(ii) **Feature Characteristics Taxonomy**, a collection of qualities to describe dataset features (e.g., data-type, maximum value). It is based on the OpenML data qualities collection(see footnote 10) and contains 55 feature characteristics.

(iii) **Evaluation Measure Taxonomy**, a collection of measures for the evaluation of ML experiments. It is based on the OpenML evaluation measures collection[12] and contains 86 evaluation measures (e.g., accuracy, recall).

(iv) The **Estimation Procedure Taxonomy**, a collection of data-splitting techniques to evaluate ML models based on the OpenML evaluation procedures collection[13] containing 5 estimation procedures (e.g., cross validation, holdout).

(v) The **Learning Method Taxonomy**, a collection of methods of learning and training in ML, based on the learning approach and the type of data they input and output. The taxonomy reuses MEX vocabulary learning method classes, learning methods found in Papers with Code ML tasks and literature review [59]. It contains 33 learning methods (e.g., supervised learning, transfer learning).

[10] https://www.openml.org/search?type=measure&measure_type=data_quality.
[11] https://paperswithcode.com/methods.
[12] www.openml.org/search?type=measure&measure_type=evaluation_measure.
[13] www.openml.org/search?type=measure&measure_type=estimation_procedure.

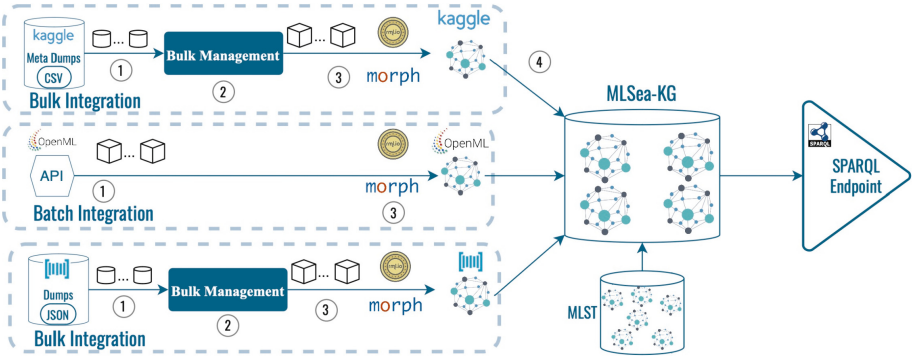

Fig. 2. MLSea-KG Construction Process Overview

(vi) The **Algorithm Taxonomy**, a collection of 2321 algorithms used in different stages of the ML process (e.g., Adam, Dropout). The taxonomy re-uses MEX vocabulary algorithm classes, the Paper with Code methods collection(see footnote 11), and ML surveys that organize ML algorithms from diverse ML fields [1,3,5,9,17,20,36,44,55,58,60–62,66].

(vii) The **Machine Learning Field Taxonomy**, a collection of areas of study in the ML domain. It is based on MEX vocabulary ML context classes and literature review [33] and contains 30 fields (e.g., computer vision, bioinformatics).

(viii) The **Task Type Taxonomy**, a collection of 1875 ML task types, defined based on the problem they try to solve (e.g., object detection, colorization). The taxonomy is based on MEX vocabulary ML context classes and task types found in OpenML tasks and Papers with Code datasets, methods, and publications.

The modular design and availability of the taxonomies aim to empower the ML community, allowing researchers and practitioners to propose new entries, suggest modifications, and redefine the hierarchy of terms via GitHub issues. This collaborative approach aims to sustain a resource that evolves in line with the dynamic landscape of ML, ensuring the taxonomies remain up-to-date.

5 MLSea-KG: Construction, Publication and Usage

We applied our MLSO ontology to create **MLSea-KG**, a KG for discovering ML data containing more than 1.44 billion triples (Fig. 2). We discuss the population and update methodology we follow for declaratively constructing and preserving MLSea-KG which is available at http://w3id.org/mlsea-kg. In this section, we showcase MLSea-KG's key statistics, availability and usage examples.

5.1 Knowledge Graph Construction Process

The KG construction process starts with the data collection step (**Step 1**, Fig. 2) for each of the examined publicly available platforms. We collected **Kaggle's** [34] data as CSV files via the **Meta Kaggle** dataset [35], a public repository containing metadata about Kaggle datasets, kernels, users, and competitions, updated on a daily basis. For **OpenML** [46], we iteratively collected all OpenML datasets, tasks, flows, runs, and their related metadata, in the form of Pandas [47] dataframes, through the **OpenML Python API** service[14]. For **Papers with Code** [50], we collected the platform's data, as JSON files, via the platform's public dump files[15] that contain metadata regarding papers, code repositories, methods, and datasets of the platform. To prepare the collected data, pre-processing modules filter out problematic data, such as faulty OpenML URLs, new-line characters in literals, and non-escaped backslashes, as well as ML pipeline-unrelated data, such as Kaggle forum messages, as we do not include platform-specific data to the MLSea-KG that are not directly related to ML.

The RDF graph is constructed by applying the MLSO and MLST to the retrieved data. We leverage the RDF Mapping Language (RML) [12,30] to map the heterogeneous data (Pandas DataFrames from OpenML, CSV files from Kaggle and JSON files from Papers with Code) to RDF triples. To define the RML mapping rules, we leverage the human-friendly serialization YARRRML [26] and the Yatter [29] tool which converts YARRRML mapping rules to RML rules.

To efficiently process large data sources, such as the Meta Kaggle and the Papers with Code metadata datasets, with RML processors we extract a batch of the complete initial dataset each time (**Step 2**, Fig. 2). We employ platform-specific custom *sampling strategies* to ensure that pertinent data are consistently grouped together within each batch. For example, when constructing an RDF graph from a batch of Kaggle datasets metadata, we also extract related Kaggle code notebooks metadata for code notebooks based on the datasets of the batch, and as a result contain useful information for the datasets' RDF graph.

We leverage Morph-KGC [2] and its extension for in-memory RDF generation [10] to declaratively generate the RDF triples for each extracted batch (**Step 3**, Fig. 2). The generated RDF batches are compressed into .gz archives, to facilitate data management and reduce the total storage. Each RDF batch is then imported into our Virtuoso triple store [15], hosted on an Ubuntu 22.04.3 LTS server with Intel(R) Xeon(R) CPU E5-2650 v2 and 128 GB RAM (**Step 4**, Fig. 2). The 3 RDF graphs, 1 for each platform, comprise the MLSea-KG.

[14] https://www.openml.org/apis.
[15] https://paperswithcode.com/about.

Table 2. MLSea-KG core entities, number of instances & triples per dataset & type.

Entity Type	Instance Count				Triples Count			
	OpenML	Kaggle	PwC	Total	OpenML	Kaggle	PwC	Total
Datasets	5.4K	277.4K	11.1K	295.9K	49.7M	4.2M	163.2K	54.1M
Tasks	47.2K	–	2.1K	49.3K	642.3K	–	8.5K	650.8K
Pipelines	16.7K	–	26.9K	43.6K	1.3M	–	1.2M	2.6M
Algorithms	–	–	2.1K	2.1K	–	–	–	19.4K
Runs	10.1M	–	26.9K	10.1M	1.36B	–	524.7K	1.36B
Models	–	–	26.9K	26.9K	–	–	214.6K	214.6K
Notebooks	–	940.4K	–	940.4K	–	9.3M	-	9.3M
Repositories	–	–	146.2K	14.2K	–	–	955.2K	955.2K
Publications	308	–	407.4K	407.4K	1.2K	–	6.8M	6.8M
Agents	1.4K	360.1K	433.5K	407K	4.3K	1.0M	1.3M	2.3M
Total	10.1M	1.5M	1.0M	12.6M	1.41B	14.5M	11.1M	**1.44B**

Continuous Integration. MLSea-KG is updated using an automatic procedure. The updated Meta Kaggle dataset and the Papers with Code dump files are downloaded and compared with their previous versions to identify new entries, while the latest data of the OpenML are downloaded through its API. Then, the same KG construction process is followed to update MLSea-KG.

Statistics of MLSea-KG. As of November 2023, MLSea-KG consists of 5.4K datasets, 47.2K ML tasks, 16.7 flows (descriptions of ML pipelines), 10.1M runs (executions, settings, and evaluations of ML experiments), 308 publications, and 1.4K agents (scientists and practitioners) from OpenML; 277K datasets, 940K ML code notebooks and 360.1K agents from Kaggle; 11.1K datasets, 2.1K ML tasks, 2.1K algorithms, 407K ML publications, 146K ML code repositories, 26.9K ML models, accompanied by experiment metadata and related evaluations, and 433K researchers from Papers with Code, all complemented with rich metadata. A total count of the instances of MLSea-KG, along with the total number of triples by instance, is exhibited in Table 2. Along with all metadata, the total triples count of MLSea-KG exceeds 1.44 billion, with the majority of the RDF triples originating from OpenML, which contains a plethora of ML-related metadata, from detailed characteristics for each dataset to numerous ML experiment configuration settings and results. Kaggle and Papers with Code focus more on generic metadata and can be, thus, represented with less RDF triples.

Availability and Maintenance. Our resource's landing page (http://w3id.org/mlsea) provides pointers to the public SPARQL endpoint of MLSea-KG (http://w3id.org/mlsea-kg), the Zenodo repository for the RDF snapshots of MLSea-KG, and the GitHub repository with the RML mapping rules for KG construction and the source code employed for metadata collection and sampling, and KG construction. The webpage summarizes how MLSea-KG was con-

structed leveraging MLSO and MLST. For at least a ten-year period, KU Leuven will *maintain* MLSea-KG, applying the continuous integration process on a monthly basis to incorporate the latest updates from all ML repositories. The KG population scripts and RML mapping rules will also be regularly updated.

5.2 Use Case Examples

The MLSea-KG allows users to search for ML datasets, tasks, implementations, models, experiments, software, and scientific works. They can view the characteristics of these components and navigate through their intricate relationships. The search over the data of the different platforms is facilitated as a user can find all relevant information with a single query. In this section, we look back at the motivating examples (Sect. 2) and demonstrate how the users can take advantage of MLSea to address the same use cases.

Example 1. *Find a dataset & its relevant code notebooks & scientific papers.* In this example, a user searches for a dataset, its code notebooks, and scientific papers over the 3 platforms simultaneously, as it was described in Sect. 2.

As opposed to this manual task, a user can retrieve the same information by leveraging MLSea. The SPARQL query of this search over MLSea-KG for the "CIFAR-10" is shown in Listing 5.1 and an excerpt of the results is shown in Table 3. The complete set of results is returned within 12 secs.

Listing 5.1. Query for CIFAR-10 dataset locations, tasks, code and publications.

```
PREFIX mlso: <http://w3id.org/mlso/>
PREFIX mls: <http://www.w3.org/ns/mls#>
#... dcterms, rdfs, dcat, prov, schema and skos are omitted
SELECT ?datasetTitle, ?datasetLocation, (GROUP_CONCAT(DISTINCT COALESCE
(?omlTask, ?pwcTask);separator=',\n') AS ?task), (GROUP_CONCAT(DISTINCT
COALESCE(?codeNotebook, ?codeRepo);separator=',\n') AS ?code),
(GROUP_CONCAT(DISTINCT ?publication; separator=',\n') AS ?introducedIn)
WHERE {
  ?dataset a mls:Dataset;
    dcat:landingPage ?datasetLocation; dcterms:title ?datasetTitle.
  OPTIONAL{?dataset dcterms:description ?label}
  OPTIONAL{?dataset rdfs:label ?label}
  FILTER(CONTAINS(?label, "CIFAR-10")||CONTAINS(LCASE(?label), "cifar-10"))
  OPTIONAL{?dataset mlso:hasRelatedSoftware ?softwareID.
    ?softwareID schema:codeRepository ?codeNotebook.}
  OPTIONAL{?taskId mls:definedOn ?dataset; prov:atLocation ?omlTask.}
  OPTIONAL{?dataset mlso:hasTaskType ?taskType.
    ?taskType skos:prefLabel ?pwcTask.}
  OPTIONAL{?dataset mlso:hasScientificReference ?publicationID.
    ?publicationID dcterms:source ?publication.}
  OPTIONAL{?dataset mlso:hasScientificReference ?publicationID.
    ?publicationID mlso:hasRelatedSoftware  ?softwareID.
    ?softwareID schema:codeRepository ?codeRepo.}
}GROUP BY ?datasetTitle ?datasetLocation
```

Table 3. Query results for CIFAR-10 dataset locations, tasks, code & publications.

dataset Title	dataset Location	task	code	introducedIn
CIFAR-FS	paperswithcode.com/dataset/cifar-fs	Few-Shot Image Classification, ...	github.com/bertinetto/r2d2	arxiv.org/pdf/1805.08136v3
CIFAR-10	kaggle.com/datasets/ayush1220/cifar10	–	kaggle.com/code/alphapii/knn-cvpr	–
CIFAR_10 small	api.openml.org/data/v1/download/16797612/CIFAR_10_small.arff	openml.org/t/294086	–	–

Listing 5.2. Query for Bagging implementations, publications & hyper-parameters.

```
PREFIX mls: <http://www.w3.org/ns/mls#>
#... dcterms, rdfs, dcat, prov, schema and skos are omitted
SELECT (STR(COALESCE(?label, ?title)) AS ?implementation),
(GROUP_CONCAT(DISTINCT ?publication; separator=',\n') AS ?publications),
(GROUP_CONCAT(DISTINCT ?algorithm; separator=',\n') AS ?algorithms),
(GROUP_CONCAT(DISTINCT ?hyperP; separator=',\n') AS ?hyperParameters)
WHERE{
  {?implemId a mls:Implementation; rdfs:label ?label.
  FILTER(CONTAINS(LCASE(?label), ''bagging''))}
  UNION{?implemId a mls:Implementation; dcterms:description ?desc;
    dcterms:title ?title. FILTER(CONTAINS(LCASE(?desc), "bagging"))}
  OPTIONAL{?pubId mlso:hasRelatedImplementation ?implemId;
    rdf:type mlso:ScientificWork; dcterms:source ?publication.}
  OPTIONAL{?implemId mls:implements ?algoId. ?algoId rdfs:label ?algorithm.}
  OPTIONAL{?implemId mls:hasHyperParameter ?hyperPID.
  ?hyperPID dcterms:description ?hyperP_desc; dcterms:title ?hyperP_title.
    BIND(CONCAT(STR(?hyperP_title),'': '', str(?hyperP_desc)) as ?hyperP)}
}GROUP BY ?label ?title
```

Example 2. *Find a pipeline & its papers, algorithms & parameters.* In this example, a user manually searches for a pipeline over the 3 platforms to find scientific papers, algorithms, and hyper-parameters, as described in Sect. 2.

As opposed to this manual task, a user can retrieve the same information by leveraging MLSea. The SPARQL query of this search over MLSea-KG is shown in Listing 5.2, where publications, algorithms, and hyper-parameters about the popular "Bagging" ensemble method are retrieved. An excerpt of the query results is shown in Table 4; the complete results are returned within 6 secs.

6 Impact of the Resource

This resource can be of interest and have impact for the ML and the Semantic Web community offering abundant prospects for future research and adoption. Practitioners can immediately use it for ML artifacts search, such as dataset

Table 4. Query results for Bagging implementations, publications & hyper-parameters.

implementation	publications	algorithms	hyperParameters
Internal node bagging	arxiv.org/pdf/1805.00215v5.pdf	Dropout	–
weka.Bagging_REPTree(22)	–	–	L: Maximum tree depth, ...
BagBERT	arxiv.org/pdf/2111.05808v1.pdf	Adam, BERT, ...	–

and model search, while researchers can leverage it for analyzing ML metadata, discover ML insights and trends, and develop ML recommendation systems.

Searchability. MLSea's immediate impact lies on the ML knowledge discovery opportunities it facilitates with the MLSea-KG and the MLST that organize ML knowledge. Researchers and practitioners can now navigate the prominent repositories (OpenML, Kaggle, and Papers with Code), by formulating precise SPARQL queries and immediately get integrated answers. They can retrieve ML datasets and view their statistical and feature characteristics, ML pipelines, their inputs, hyper-parameter settings and results, ML tasks and related implementations that attempt to solve them, ML peer-reviewed publications, code notebooks, and ML models, precisely tailored to their needs.

This semantic layer has the potential to accelerate the progress of ML projects by facilitating more efficient searches through previous works and enabling their reuse. For example, users can reuse or augment datasets, identify optimal hyper-parameter settings for a specific task or leverage pre-trained models, via transfer learning. Likewise, the integration of the taxonomies (MLST) with the MLSea-KG can enhance the search capabilities over MLSea-KG. For instance, users can query MLSea-KG for a task with a specific type from the ML Task Type taxonomy.

MLSea does not only facilitate individual platform searches, but also enables combined searches over platforms. The diverse data that the resource provides offer new capabilities for fostering a more comprehensive understanding of ML experiments, by connecting fragmented information. For instance, discovering a dataset from Papers with Code and subsequently locating relevant code notebooks in Kaggle enhances cross-platform connectivity, promoting a deeper and more complete understanding of the available ML resources.

Reproducibility. Utilizing MLSea to model and store ML pipelines significantly enhances documentation and has the potential to increase the repeatability, reproducibility, traceability, and explainability of ML pipelines. It promotes a standardized approach to sharing experiments, enhancing knowledge exchange among practitioners and interoperability amongst ML online repositories. Simultaneously, MLST function as a dynamic thesaurus, offering a structured repository for organizing ML knowledge in a modular way.

Data Analysis. By leveraging MLSea-KG, new data analysis and insights opportunities emerge. Through innovative techniques, such as graph embeddings, users can derive significant insights, such as hidden patterns and trends, from connections prevalent in the graph. Users can investigate relationships between dataset characteristics, the impact of different hyper-parameter settings on model performance, or discover correlations between specific algorithms and successful experiments. These potential explorations can lead to a better understanding of best practices, optimizations, and success factors within ML projects.

Recommendations. MLSea provides an extensive data source for ML recommendation systems to be based on by harnessing past successful experiments and user requirements to recommend suitable datasets, pre-trained models,

hyper-parameter settings, and algorithms. These recommendation systems could complement AutoML systems in providing solutions to ML practitioners. For instance, a recommendation system could analyze experiment outcomes, and propose related datasets, effective models or algorithms for a given context.

7 Conclusions and Future Work

This work leverages semantic web technologies to meticulously model diverse ML information and encompass a large corpus of ML datasets, experiments, software and scientific publications. The introduced MLSO and MLST extend and combine past ontologies and W3C recommendations to provide a versatile schema that describes different facets of the ML process.

We applied this schema to 3 prominent ML platforms (OpenML, Kaggle, and Papers with Code), using their metadata related to ML processes to generate RDF graphs, ultimately creating MLSea-KG, a large-scale KG for ML, boasting over 1.44 billion RDF triples. We discuss MLSea-KG's construction and update, its characteristics, use cases, and potential impact for the ML community, highlighting its usefulness for practitioners and researchers on a variety of levels.

In the future, we plan to continue updating the resources, integrating the latest data of the examined platforms. We also plan to extend our resources, by modeling and incorporating metadata from additional ML repositories, thus encompassing an even broader picture of the ML landscape. Furthermore, we aim to enhance specificity in representing deep learning components and experiments, fostering a more detailed layer of deep learning semantic descriptions. Finally, we intend to explore further potential of our resource, examining its use as a source for data analysis, insights discovery, and ML recommendation systems.

Acknowledgements. This research was partially supported by Flanders Make (REXPEK project), the strategic research centre for the manufacturing industry and the Flanders innovation and entrepreneurship (VLAIO – KG3D project).

References

1. AlMahamid, F., Grolinger, K.: Reinforcement learning algorithms: an overview and classification. In: 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1–7 (2021). https://doi.org/10.1109/CCECE53047.2021.9569056
2. Arenas-Guerrero, J., Chaves-Fraga, D., Toledo, J., Pérez, M.S., Corcho, O.: Morph-KGC: scalable knowledge graph materialization with mapping partitions. Semant. Web (2022). https://doi.org/10.3233/SW-223135
3. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017). https://doi.org/10.1109/MSP.2017.2743240
4. Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., Vidal, M.E.: Towards a knowledge graph for science. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, pp. 1–6 (2018). https://doi.org/10.1145/3227609.3227689

5. Bielza, C., Larrañaga, P.: Discrete Bayesian network classifiers: a survey. ACM Comput. Surv. **47**(1) (2014). https://doi.org/10.1145/2576868
6. Breit, A., et al.: Combining machine learning and semantic web: a systematic mapping study. ACM Comput. Surv. **55**(14s) (2023). https://doi.org/10.1145/3586163
7. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.99 (2014). http://xmlns.com/foaf/spec/
8. Castellano, G., Digeno, V., Sansaro, G., Vessio, G.: Leveraging knowledge graphs and deep learning for automatic art analysis. Knowl.-Based Syst. **248**, 108859 (2022). https://doi.org/10.1016/j.knosys.2022.108859
9. Charbuty, B., Abdulazeez, A.: Classification based on decision tree algorithm for machine learning. J. Appl. Sci. Technol. Trends **2**(01), 20–28 (2021). https://doi.org/10.38094/jastt20165
10. Dasoulas, I., Chaves-Fraga, D., Garijo, D., Dimou, A.: Declarative RDF construction from in-memory data structures with RML. In: Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference ESWC 2023, vol. 1613, p. 0073 (2023)
11. Debattista, J., Lange, C., Auer, S.: daQ, an ontology for dataset quality information. In: LDOW (2014)
12. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Bizer, C., Heath, T., Auer, S., Berners-Lee, T. (eds.) Proceedings of the 7th Workshop on Linked Data on the Web. CEUR Workshop Proceedings, vol. 1184. CEUR (2014)
13. Draw.io: Security-first diagramming for teams. https://www.drawio.com. Accessed 28 Nov 2023
14. Ekaputra, F.J., et al.: Describing and organizing semantic web and machine learning systems in the SWeMLS-KG. In: Pesquita, C., et al. (eds.) The Semantic Web. ESWC 2023. LNCS, vol. 13870, pp. 372–389. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33455-9_22
15. Erling, O., Mikhailov, I.: Virtuoso: RDF support in a native RDBMS. In: de Virgilio, R., Giunchiglia, F., Tanca, L. (eds.) Semantic Web Information Management, pp. 501–519. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04329-1_21
16. Esteves, D., et al.: MEX vocabulary: a lightweight interchange format for machine learning experiments. In: Proceedings of the 11th International Conference on Semantic Systems, pp. 169–176. SEMANTICS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2814864.2814883
17. Fahad, A., et al.: A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Trans. Emerg. Top. Comput. **2**(3), 267–279 (2014). https://doi.org/10.1109/TETC.2014.2330519
18. Färber, M., Lamprecht, D.: Linked papers with code: the latest in machine learning as an RDF knowledge graph. In: ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference, 6–10 November 2023, Athens, Greece (2023). https://doi.org/10.48550/arXiv.2310.20475
19. Färber, M., Lamprecht, D., Krause, J., Aung, L., Haase, P.: SemOpenAlex: the scientific landscape in 26 billion RDF triples. In: Payne, T.R., et al. (eds.) The Semantic Web – ISWC 2023. ISWC 2023. LNCS, vol. 14266, pp. 94–112. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47243-5_6, https://doi.org/10.48550/arXiv.2308.03671

20. Fürnkranz, J., Kliegr, T.: A brief overview of rule learning. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) RuleML 2015. LNCS, vol. 9202, pp. 54–69. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21542-6_4
21. Garijo, D.: WIDOCO: a wizard for documenting ontologies. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 94–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_9
22. Garijo, D., Osorio, M., Khider, D., Ratnakar, V., Gil, Y.: OKG-Soft: an open knowledge graph with machine readable scientific software metadata. In: 15th International Conference on eScience (eScience), pp. 349–358 (2019). https://doi.org/10.1109/eScience.2019.00046
23. Gundersen, O.E., Shamsaliei, S., Isdahl, R.J.: Do machine learning platforms provide out-of-the-box reproducibility? Futur. Gener. Comput. Syst. **126**, 34–47 (2022). https://doi.org/10.1016/j.future.2021.06.014
24. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. Recommendation, World Wide Web Consortium (W3C), March 2013. https://www.w3.org/TR/sparql11-query/
25. Helal, A., Helali, M., Ammar, K., Mansour, E.: A demonstration of KGLac: a data discovery and enrichment platform for data science. Proc. VLDB Endow. **14**(12), 2675–2678 (2021). https://doi.org/10.14778/3476311.3476317
26. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative rules for linked data generation at your fingertips! In: The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, 3–7 June 2018, Revised Selected Papers 15 (2018)
27. Hugging Face: Hugging Face – The AI community building the future. https://huggingface.co. Accessed 28 Nov 2023
28. Hutson, M.: Artificial intelligence faces reproducibility crisis (2018). https://doi.org/10.1126/science.359.6377.725
29. Iglesias-Molina, A., Chaves-Fraga, D., Dasoulas, I., Dimou, A.: Human-Friendly RDF graph construction: which one do you chose? In: Garrigós, I., Murillo Rodríguez, J.M., Wimmer, M. (eds.) Web Engineering. ICWE 2023. LNCS, vol. 13893, pp. 262–277. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34444-2_19
30. Iglesias-Molina, A., et al.: The RML ontology: a community-driven modular redesign after a decade of experience in mapping heterogeneous data to RDF. In: Payne, T.R., et al. (eds.) The Semantic Web – ISWC 2023. ISWC 2023. LNCS, vol. 14266, pp. 152–175. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47243-5_9
31. Ismaeil, Y., Stepanova, D., Tran, T.K., Saranrittichai, P., Domokos, C., Blockeel, H.: Towards neural network interpretability using commonsense knowledge graphs. In: Sattler, U., et al. (eds.) The Semantic Web – ISWC 2022. ISWC 2022. LNCS, vol. 13489, pp. 74–90. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19433-7_5
32. Ison, J., et al.: EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics **29**(10), 1325–1332 (2013). https://doi.org/10.1093/bioinformatics/btt113
33. Jordan, M.I., Mitchell, T.M.: Machine learning: trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015). https://doi.org/10.1126/science.aaa8415
34. Kaggle: Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com. Accessed 28 Nov 2023

35. Kaggle: Meta Kaggle - Kaggle's public data on competitions, users, submission scores, and kernels. https://www.kaggle.com/datasets/kaggle/meta-kaggle. Accessed 28 Nov 2023
36. Kaur, J., Madan, N.: Association rule mining: a survey. Int. J. Hybrid Inf. Technol. **8**(7), 239–242 (2015)
37. Keet, C.M., et al.: The data mining optimization ontology. J. Web Semant. **32**, 43–53 (2015). https://doi.org/10.1016/j.websem.2015.01.001
38. Lebo, T., et al.: PROV-O: The PROV Ontology. Recommendation, World Wide Web Consortium (W3C), April 2013. https://www.w3.org/TR/prov-o/
39. Li, L., et al.: Real-world data medical knowledge graph: construction and applications. Artif. Intell. Med. **103**, 101817 (2020). https://doi.org/10.1016/j.artmed.2020.101817
40. Maali, F., Erickson, J.: Data Catalog Vocabulary (DCAT). Recommendation, World Wide Web Consortium (W3C), January 2014. https://www.w3.org/TR/vocab-dcat/
41. Malone, J., et al.: The software ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation. J. Biomed. Semant. **5**(1), 1–13 (2014). https://doi.org/10.1186/2041-1480-5-25
42. McGuinness, D.L., Van Harmelen, F., et al.: Owl web ontology language overview. W3C Recommendation (2004)
43. Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System Reference. Recommendation, World Wide Web Consortium (W3C), August 2009. https://www.w3.org/TR/skos-reference/
44. Moradi, R., Berangi, R., Minaei, B.: A survey of regularization strategies for deep models. Artif. Intell. Rev. **53**, 3947–3986 (2020). https://doi.org/10.1007/s10462-019-09784-7
45. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015). https://doi.org/10.1145/2757001.2757003
46. OpenML: OpenML: a worldwide machine learning lab. https://www.openml.org
47. pandas: Pandas - A fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. https://pandas.pydata.org. Accessed 28 Nov 2023
48. Panov, P., Džeroski, S., Soldatova, L.: OntoDM: an ontology of data mining. In: 2008 IEEE International Conference on Data Mining Workshops, pp. 752–760. IEEE (2008). https://doi.org/10.1109/ICDMW.2008.62
49. Panov, P., Soldatova, L., Džeroski, S.: OntoDM-KDD: ontology for representing the knowledge discovery process. In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (eds.) DS 2013. LNCS (LNAI), vol. 8140, pp. 126–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40897-7_9
50. Papers with Code: Papers With Code: The latest in Machine Learning. https://paperswithcode.com. Accessed 28 Nov 2023
51. Peroni, S., Shotton, D.: FaBiO and CiTO: ontologies for describing bibliographic resources and citations. J. Web Semant. **17**, 33–43 (2012). https://doi.org/10.1016/j.websem.2012.08.001
52. Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., García-Castro, R.: LOT: an industrial oriented ontology engineering framework. Eng. Appl. Artif. Intell. **111**, 104755 (2022). https://doi.org/10.1016/j.engappai.2022.104755
53. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy Pitfall Scanner!): an on-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**(2), 7–34 (2014). https://doi.org/10.4018/ijswis.2014040102

54. Publio, G.C., et al.: ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies. arXiv preprint arXiv:1807.05351 (2018). https://doi.org/10.48550/arXiv.1807.05351
55. Ravishankar, N., Vijayakumar, M.: Reinforcement learning algorithms: survey and classification. Indian J. Sci. Technol. **10**(1), 1–8 (2017). https://doi.org/10.17485/ijst/2017/v10i1/109385
56. Ristoski, P., Paulheim, H.: Semantic web in data mining and knowledge discovery: a comprehensive survey. J. Web Semant. **36**, 1–22 (2016). https://doi.org/10.1016/j.websem.2016.01.001
57. Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S., Sontag, D.: Learning a health knowledge graph from electronic medical records. Sci. Rep. **7**(1), 5994 (2017). https://doi.org/10.1038/s41598-017-05778-z
58. Ruder, S.: An overview of gradient descent optimization algorithms (2017).https://doi.org/10.48550/arXiv.1609.04747
59. Sah, S.: Machine learning: a review of learning types. Int. Res. J. Mod. Eng. Technol. Sci. (2020). https://doi.org/10.20944/preprints202007.0230.v1
60. Saxena, A., et al.: A review of clustering techniques and developments. Neurocomputing **267**, 664–681 (2017). https://doi.org/10.1016/j.neucom.2017.06.053
61. Sharma, H., Kumar, S., et al.: A survey on decision tree algorithms of classification in data mining. Int. J. Sci. Res. (IJSR) **5**(4), 2094–2097 (2016)
62. Shrestha, A., Mahmood, A.: Review of deep learning algorithms and architectures. IEEE Access **7**, 53040–53065 (2019). https://doi.org/10.1109/ACCESS.2019.2912200
63. Soldatova, L.N., King, R.D.: An ontology of scientific experiments. J. R. Soc. Interface **3**(11), 795–803 (2006). https://doi.org/10.1098/rsif.2006.0134
64. Souza, R., et al.: Provenance data in the machine learning lifecycle in computational science and engineering. In: 2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), pp. 1–10. IEEE (2019). https://doi.org/10.1109/WORKS49585.2019.00006
65. TensorFlow: An end-to-end open source machine learning platform for everyone. https://www.tensorflow.org. Accessed 28 Nov 2023
66. Tian, Y., Zhang, Y.: A comprehensive survey on regularization strategies in machine learning. Inf. Fusion **80**, 146–166 (2022). https://doi.org/10.1016/j.inffus.2021.11.005
67. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases: a new way to share, organize and learn from experiments. Mach. Learn. **87**, 127–158 (2012). https://doi.org/10.1007/s10994-011-5277-0
68. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explor. Newsl. **15**(2), 49–60 (2014). https://doi.org/10.1145/2641190.2641198
69. Venkataramanan, R., Tripathy, A., Foltin, M., Yip, H.Y., Justine, A., Sheth, A.: Knowledge graph empowered machine learning pipelines for improved efficiency, reusability, and explainability. IEEE Internet Comput. **27**(1), 81–88 (2023). https://doi.org/10.1109/MIC.2022.3228087
70. Villanueva Zacarias, A.G., Reimann, P., Weber, C., Mitschang, B.: AssistML: an approach to manage, recommend and reuse ML solutions. Int. J. Data Sci. Anal. 1–25 (2023). https://doi.org/10.1007/s41060-023-00417-5
71. Weibel, S.L., Koch, T.: DCMI metadata terms. Technical report, Dublin Core Metadata Initiative (2012). http://dublincore.org/documents/dcmi-terms/

72. Zheng, Z., et al.: Executable knowledge graphs for machine learning: a Bosch case of welding monitoring. In: Sattler, U., et al. (eds.) The Semantic Web – ISWC 2022. ISWC 2022. LNCS, vol. 13489, pp. 791–809. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19433-7_45
73. Zhou, B., et al.: SemML: facilitating development of ML models for condition monitoring with semantics. J. Web Semant. **71**, 100664 (2021). https://doi.org/10.1016/j.websem.2021.100664