Article

An Industry-Ready Machine Learning Ontology

Bernhard G Humm *

Posted Date: 19 November 2025

doi: 10.20944/preprints202511.1420.v1

Keywords: ontology; machine learning; reasoning; ontology alignment

 doi:10.20944/preprints202511.1420.v1

Article

An Industry-Ready Machine Learning Ontology

Bernhard G. Humm

Darmstadt University of Applied Sciences, Schöfferstr. 3, 64295 Darmstadt, Germany; bernhard.humm@h-da.de

Featured Application

The ML Ontology presented in this article is the information backbone of a data science platform used for training machine learning models from datasets. Features include a training configuration wizard and interactive help.

Abstract

This article presents an industry-ready ontology for the machine learning domain, which is named "ML Ontology". ML ontology is comprehensive, provides good performance and is extensible and adaptable. While based on lightweight modelling languages, ML ontology provides novel features including built-in queries and quality assurance, as well as sophisticated reasoning. Its industry-readiness is demonstrated by benchmarks as well as two use case implementations within a data science platform.

Keywords: ontology; machine learning; reasoning; ontology alignment

1. Introduction

Why still work on ontologies while generative artificial intelligence (AI), in particular large language models (LLM) today allow solving problems for which ontologies have been used traditionally? The reason is that there are still numerous use cases, also in industry, in which the explicit specification of concepts and their relationships are most useful. Ontologies [1] serve exactly this purpose. Many ontologies have been developed for the life sciences domain, e.g., Gene Ontology (GO)[1], SNOMED CT[2] and MeSH (Medical Subject Headings)[3]. However, in the AI domain and the field of ML in particular, there are only a handful of ontologies available (e.g., [2–5]). In this article, an ontology for the ML field is presented which is named *ML Ontology*. ML Ontology is based on existing work and expands on the following aspects:

1. **Volume**: ML Ontology is the most comprehensive ontology for the domain of ML that the author is aware of.
2. **Performance**: ML Ontology allows high-performance query access suitable for industry applications.
3. **Balance between simplicity and expressiveness**: It is based on simple technology particularly suited for industry use, while at the same time offers expressive power, including sophisticated reasoning where needed.
4. **Extensibility and adaptability**: ML Ontology is modularised and can be easily extended and adapted to differing use cases.
5. **Built-in quality management**: ML Ontology comprises built-in quality checks that can be adapted use-case-specifically in order to ensure quality requirements from industry.

[1] https://geneontology.org
[2] https://www.snomed.org
[3] https://meshb.nlm.nih.gov

 doi:10.20944/preprints202511.1420.v1

6. **Standards**: ML Ontology is solely based on Semantic Web (SW)[4] standards like RDF[5], RDFS[6] and SPARQL[7] and may also well be deployed using industry-ready Labeled Property Graph (LPG) databases like Neo4j[8].
7. **Public availability**: ML Ontology is published open source[9] under an MIT license.

Many of those properties are crucial for using ontologies in industry applications, including sufficient volume, performance, expressiveness, extensibility and adaptability, quality and conformance to standards. In this article, the term *industry-ready* is used for ontologies that exhibit those properties.

ML Ontology has been developed over a period of more than five years by more than 15 different knowledge engineers with knowledge in the ML field. In the current version it consists of ca. 5,000 RDF triples. The following two use cases use ML Ontology, and demonstrate its industry readiness:

1. **Configuration wizard**: ML Ontology serves as the information backbone for a data science platform. Its usage in the configuration wizard for expert configurations of ML training runs is demonstrated.
2. **Interactive help**: ML Ontology is used as the information source for the interactive help system of the data science platform, helping users understand complex terminology in machine learning.

The main contribution of this article is an industry-ready ontology for the ML field, which expands on previous work with built-in quality management and a balance between simplicity and expressiveness, leading to extensibility, adaptability and high-performance access.

The remainder of this article is structured as follows. Section 2 references related work. Section 3 is the core of this article and explains the architecture of ML Ontology. In Section 4, two use cases are introduced which demonstrate successful application of ML ontology. Section 5 discusses findings and Section 6 concludes this article.

2. Related Work

A commonly cited definition of ontology is "an explicit specification of a conceptualisation" by Gruber [1]. Gruber furthermore explains conceptualisation as "the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them". This definition is broad enough to include formal upper-level ontologies but also knowledge graphs, thesauri and other forms of knowledge representation.

The world-wide web consortium (W3C)[10] has standardised modelling languages for ontologies under the Semantic Web (SW) umbrella. RDF (Resource Description Framework) allows specifying structured information as subject–predicate–object triples in a machine-readable and interoperable way. RDFS (RDF Schema) is an extension of RDF that provides basic vocabulary for defining classes, properties, and relationships to build simple ontologies. OWL (Web Ontology Language)[11] is a powerful ontology language built on RDF/RDFS that allows for rich, logic-based modelling of complex relationships and constraints between concepts. SPARQL (SPARQL Protocol and RDF Query Language) is a query language designed to retrieve and manipulate data stored in RDF format, similar in syntax to SQL but tailored for graph-based data.

W3C's standardisation efforts have spawned the definition of numerous widely-accepted ontology schemas. SKOS (Simple Knowledge Organization System)[12] is a W3C standard that provides a lightweight RDF-based framework for representing controlled vocabularies like thesauri, taxonomies,

[4] https://www.w3.org
[5] https://www.w3.org/RDF
[6] https://www.w3.org/TR/rdf-schema
[7] https://www.w3.org/TR/sparql11-overview
[8] https://neo4j.com
[9] https://github.com/hochschule-darmstadt/MetaAutoML/blob/main/controller/managers/ontology/ML_Ontology.ttl
[10] https://www.w3.org
[11] https://www.w3.org/TR/owl-features
[12] https://www.w3.org/2004/02/skos

 doi:10.20944/preprints202511.1420.v1

and classification schemes. SPIN[13] is a W3C Member Submission that has become the de-facto industry standard to represent SPARQL queries and rules.

Labeled Property Graphs (LPG) [6] are also graph-based data models like RDF/RDFS. One prominent LPG database system is Neo4j, which is widely used in industrial practice. GQL (Graph Query language)[14] is a standarisation effort by ISO (International Standardization Organization)[15], which is gradually being adopted by graph database systems such as Neo4j.

The term *knowledge graph* has been used in recent years[16]. For example, Google Knowledge Graph uses simple standardised schemas like Schema.org[17]. Schema.org specifies a limited number of classes (currently about 800) such as persons, organisations, locations, products, events, etc. Simple standardised modelling languages including RDFa, Microdata and JSON-LD may be used to specify concrete entities. ML Ontology can be categorised as a knowledge graph.

In own previous work [8] we discuss various flavours of ontologies that can be subsumed under Gruber's definition referring to them as *lightweight* versus *heavy-weight*. We consider knowledge graphs as lightweight since simple modelling languages and schemas are used. Lightweight ontologies are relatively easy to use and are used frequently in industry practice, e.g., by all major internet search engine providers and tech companies, including Google, Amazon, eBay, Facebook, IBM, LinkedIn and Microsoft [7].

In contrast, heavy-weight ontologies use modelling languages with much higher semantic expressiveness, such as OWL[18] or F-logic [9]. Such ontology modelling languages allow knowledge engineers to express reasoning, logical operators, quantifiers, etc.

The architecture of ML Ontology is based on own previous work [10] which present an extensible approach to multi-level ontology modelling. In particular, a lightweight mechanism for using inferencing in an extensible and use-case-specific way is presented. This lightweight approach is also employed for ML Ontology to balance simplicity and expressiveness.

There are relatively few ontologies for the ML domain that are published, maintained and publicly available. *ML Schema*[19] [2] has been developed by the W3C Machine Learning Schema Community Group. It is a top-level ontology for describing ML algorithms, datasets, models, and experiments. It has been used for mapping OntoDM, DMOP, Exposé and MEX Vocabulary. ML Ontology is also linked to ML-Schema. *OntoDM-core* [3] is a comprehensive ontology for data mining and ML, aligned with top-level ontologies such as BFO and OBI. In contrast to ML Ontology, OntoDM-core focusses on ML experiments. *DMOP (Data Mining OPtimization Ontology)* [4] provides representations of ML workflows, focussing on optimisation and meta-mining. DMOP does not seem to be publicly available. *Exposé Ontology* [5] is used to describe and reason about ML experiments. It does not seem to be publicly available. *MEX Vocabulary*[20] [4] is a lightweight vocabulary for describing ML experiments and their configurations.

Wikidata[21] is a large-scale, general-purpose knowledge graph. It is the information backbone of Wikipedia and serves as a central source of open data. Wikidata also includes numerous entries for the ML domain. ML Ontology links ML concepts to Wikidata.

---

13 https://spinrdf.org
14 https://www.gqlstandards.org
15 https://www.iso.org
16 Although the term knowledge graph has been already used since 1972, the current use of the phrase stems from the 2012 announcement of the Google Knowledge Graph [7]
17 https://schema.org
18 https://www.w3.org/TR/owl-features
19 https://ml-schema.github.io/documentation
20 https://github.com/mexplatform/mex-vocabulary/tree/master
21 https://www.wikidata.org

 doi:10.20944/preprints202511.1420.v1

3. Ontology Architecture

3.1. Modelling Approach

To ensure industry readiness, ML ontology uses a lightweight modelling approach and is implemented as a knowledge graph. The modelling language used is RDF/RDFS. ML Ontology includes limited number of RDFS classes (ca. 10) and a large number of instances (ca. 700) representing ML concepts. Some SKOS properties like `skos:prefLabel`, `skos:altLabel` or `skos:broader` are used for attributes and relationships. SPARQL is used as query language and SPARQL Update for reasoning; SPIN is used for storing SPARQL queries and insert statements as part of the ontology.

3.2. ML Ontology Schema

Figure 1 shows the schema of ML Ontology as a UML diagram[22]. ML Ontology consists of four modules:

1. **Classes**: Definition of all classes and properties to be used in the ontology;
2. **Instances**: Instances of the specified classes with all their properties that model ML concepts and their relationships;
3. **Inference rules**: Forward-chaining inference rules may be applied in a use-case-specific manner to enhance the expressiveness of the ontology;
4. **QA**: Quality assurance (QA) checks can be used in a use-case-specific manner to ensure integrity of the ontology;
5. **Queries**: Sample queries that may be used in applications utilising the ontology.

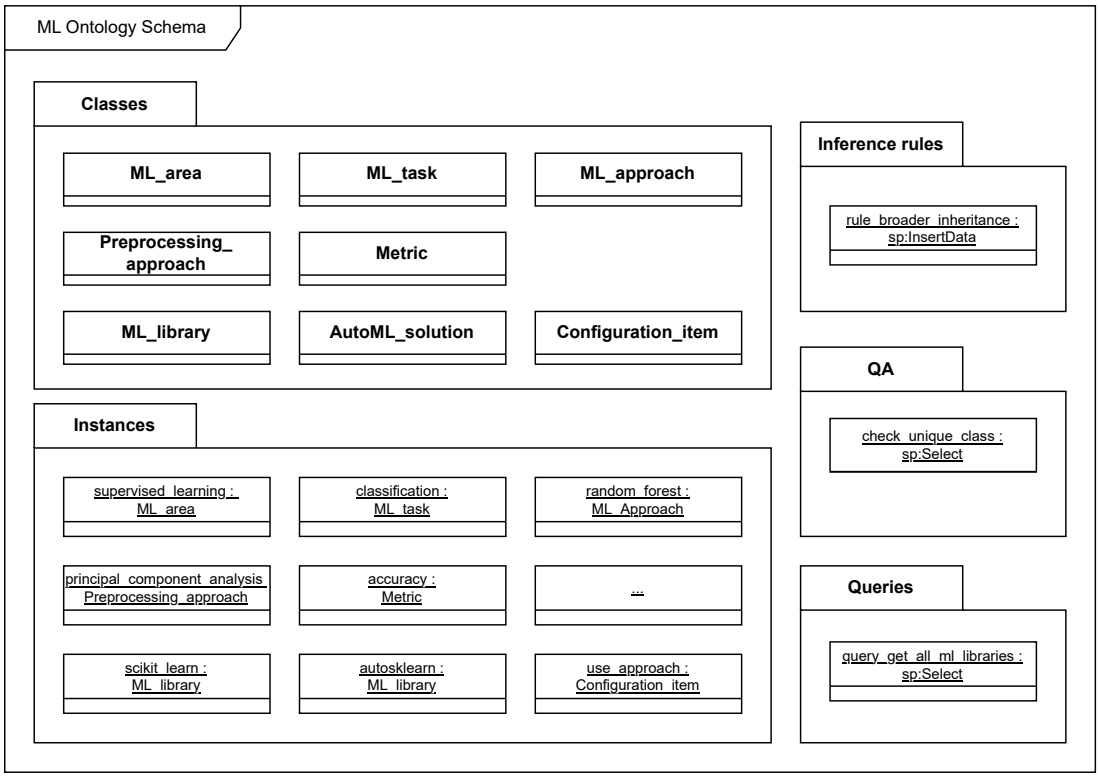

Figure 1. ML Ontology schema

The main classes of ML Ontology are as follows.

- `ML_area` with instances for supervised learning, unsupervised learning and reinforcement learning;
- `ML_task` with ca. 40 instances, e.g., for classification, regression, and clustering;

[22] https://www.omg.org/spec/UML

 doi:10.20944/preprints202511.1420.v1

- `ML_approach` with ca. 120 instances, e.g., for artificial neural network, random forest, and support vector machine;
- `Preprocessing_approach` with ca. 10 instances, e.g., for principal component analysis and kernel approximation;
- `Metric` with ca. 120 instances, e.g., for accuracy, f-measure and mean squared error;
- `ML_library` with ca. 30 instances, e.g., for Scikit-learn, Tensorflow, and PyTorch;
- `AutoML_solution` with ca. 30 instances, e.g., for AutoSklearn, TPOT, and Auto-PyTorch;
- `Configuration_item` with ca. 240 instances, e.g., for ensemble size, use approach, and metric.

In addition, there are ca. 15 predefined sample queries (e.g., get all dataset types), 5 quality assurance checks (e.g., check unique class) and one inference rule (broader inheritance). Classes and instances are explained in more detail in the following sections.

3.3. Ontology Classes

Figure 2 shows the main classes of ML Ontology as a UML diagram. The classes are organised in two modules:

1. **ML concepts** formalise terminology in the ML domain: concepts and their interdependencies;
2. **ML implementations** specify ML libraries, AutoML solutions and their concrete configuration options.

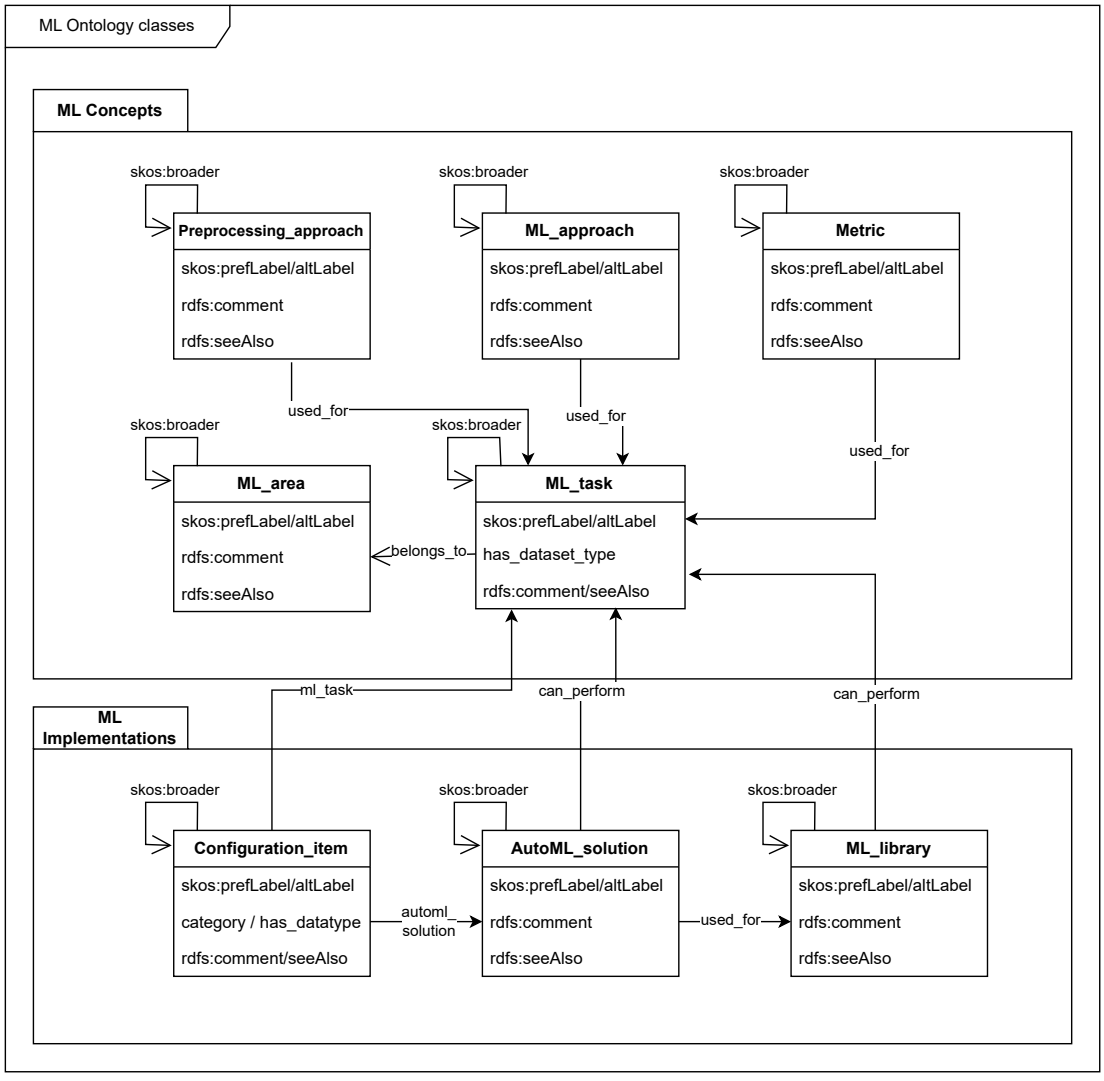

Figure 2. ML Ontology classes

 doi:10.20944/preprints202511.1420.v1

Instances of all classes are specified by a set of common properties:

1. `skos:prefLabel`: the preferred label of the individual, e.g., "support vector machine". This label is used for the IRI of the individual, e.g., `support_vector_machine`;
2. `skos:altLabel`: alternative labels or acronyms, e.g., "SVN";
3. `rdfs:comment`: short description of the individual, e.g., "Set of methods for supervised statistical learning";
4. `rdfs:seeAlso`: ontology linking, e.g., the Wikidata entry for support vector machine <https://www.wikidata.org/entity/Q282453https://www.wikidata.org/entity/Q282453>.

In addition, there are class-specific properties and properties that link instances of difference classes:

1. `used_for` links ML approaches, preprocessing approaches and metrics to ML tasks they can be used for, e.g., the ML approach support vector machine can be used for ML tasks classification and regression.
2. `belongs_to` links ML tasks to ML areas, e.g., the ML task classification belongs to the ML area supervised learning.
3. `can_perform` links instances of ML library or AutoML solution to ML tasks, e.g., ML library scikit-learn can perform ML task classification.
4. `ml_task` links configuration items to ML tasks.
5. `skos:broader` is used for modelling taxonomies between instances of the same class, e.g., the broader term of tabular regression is regression.

3.4. Ontology Alignment

Ontology alignment allows identifying correspondences between concepts in different ontologies to enable semantic interoperability across heterogeneous data sources. It reduces ambiguity, enhances reusability of knowledge, and allows applications to make use of linked data.

In ML Ontology, classes have been aligned with concepts from the ML Schema Core Specification[23] and instances with Wikidata entries via `rdfs:seeAlso`. See Tables 1 and 2.

Table 1. Alignment between ML Ontology and ML Schema

ML Ontology	ML Schema
ML_task	Task
ML_approach	Algorithm
Preprocessing_approach	Algorithm
Metric	EvaluationMeasure
ML_library	Implementation
AutoML_solution	Implementation
Configuration_item	HyperParameter

Table 2. Alignment between ML Ontology and Wikidata

Class	Instance	Wikidata ID	Wikidata Label
ML_area	supervised_learning	Q334384	supervised learning
ML_task	classification	Q13582682	classification
ML_approach	artificial_neural_network	Q192776	artificial neural network
Preprocessing_approach	principal_component_analysis	Q2873	principal component analysis
Metric	accuray	Q272035	accuracy and precision
ML_library	scikit_learn	Q1026367	scikit-learn
AutoML_solution	autosklearn	Q120703207	auto-sklearn

[23] Namespace <https://www.w3.org/ns/mls#>

 doi:10.20944/preprints202511.1420.v1

3.5. Ontology Instances

Figure 3 shows an example subset of ontology instances, their attributes and relationship as a UML diagram. RDF triples with their subject–predicate–object structure are oriented at natural languages. A triple in Turtle notation like

```
:regression :belongs_to :supervised_learning .
```

can be read as "The ML task regression belongs to the ML area supervised learning." In the following, the various ML concepts depicted in Figure 3 are explained in order to to illustrate the kind of knowledge being represented in ML Ontology. The areas explained are indicated with numbers in red circles, starting in the centre of the diagram.

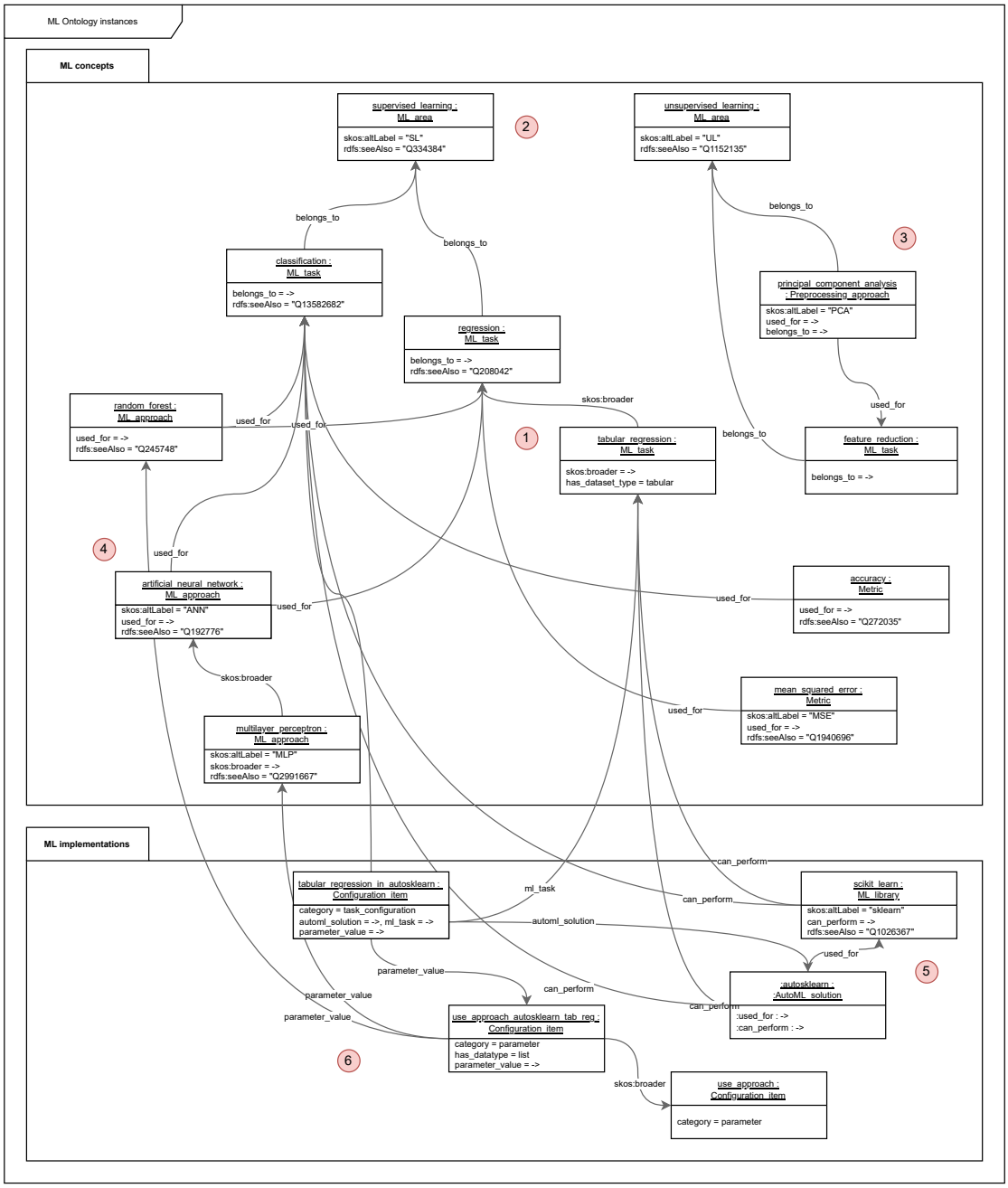

Figure 3. ML Ontology instances

(1) ML task tabular regression is a specialisation of the concept regression, hence the broader relationship. The specified dataset type is tabular. `tabular` is an instance of the auxiliary class `Enum`

 doi:10.20944/preprints202511.1420.v1

(not depicted in Figure 2), which allows formalising categorised named vocabularies. In this case, the category is dataset type and instances are tabular, text, etc. Where there are Wikidata entries for respective concepts, there are links to respective entries, e.g., for regression the entry Q208042, https://www.wikidata.org/entity/Q208042.

(2) The ML task regression belongs to ML area supervised learning (acronym "SL", Wikidata ID Q334384). Another ML task belonging to supervised learning is classification. The metric accuracy can be used for classification and the metric mean squared error can be used for regression.

(3) An ML task belonging to ML area unsupervised learning is feature reduction, just like the preprocessing approach principal component analysis (PCA).

(4) The ML approaches random forest and artificial neural network (ANN) can both be used for ML tasks classification and regression. The ML approach multilayer perceptron (MLP) is a specialisation of artificial neural network, hence the broader relationship.

(5) Auto-Sklearn is an AutoML solution that can be used for generating ML models for ML library Scikit-learn. Both can perform ML tasks classification and regression.

(6) We show exemplary a configuration item tabular regression in autosklearn. It configures multiple parameter values, one of which is use approach for configuring the ML approaches that Auto-Sklearn supports (here random forest and artificial neural network). The broader term is use approach, necessary for aligning the parameter configurations between various AutoML solutions.

3.6. Ontology Queries

Traditionally, ontologies and queries are separated. Ontologies are published, e.g., as open data, whereas SPARQL queries are developed as part of applications using the data. However, we argue that ontologies and queries belong together because predicates and classes used in queries depend on their definitions in the ontology. Therefore, sample SPARQL queries are published as part of ML Ontology, providing application developers wishing to access ML Ontology a starting point for their implementation.

SPIN[24] is employed for storing SPARQL queries within ML Ontology. Consider the following statements for specifying a SPARQL query which gets all ML tasks and the respective ML areas they belong to, e.g., classifcation belongs to supervised learning.

```
:query_get_all_ml_tasks a sp:Select;
    sp:text '''
        PREFIX:<http://h-da.de/ml-ontology/>
        SELECT ?t ?a
        WHERE {
            ?t a :ML_task;
            :belongs_to ?a.
    }'''.
```

The class `sp:Select`[25] is instantiated and used with the property `sp:text` for specifying the SPARQL query.

3.7. Quality Assurance

Like many other ontologies, ML Ontology has been curated by a number of different knowledge engineers over a longer period of time. As in software development, architectural integrity is an important aspect in ontology development. In software development, regression testing is a standard approach for ensuring quality over development time. We consider the same to be applicable for ontology development. Therefore, a set of quality checks have been developed to ensure adherence to integrity requirements. Examples of such requirements are:

[24] https://spinrdf.org
[25] Namespace http://spinrdf.org/spin#

 doi:10.20944/preprints202511.1420.v1

1. All subjects or objects in RDF triples must be instances of exactly one class.
2. All predicates used in RDF triples must be be declared as properties.

SPARQL queries have been implemented for checking violations of such quality requirements and store them as part of ML Ontology using SPIN. Below, a simple sample SPARQL query is shown which filters individuals that are instances of more than one class.

```
SELECT DISTINCT ?subject ?class1 ?class2
WHERE {
        ?subject a ?class1 .
        ?subject a ?class2 .
        FILTER(?class1 != ?class2)
}
```

The expected behaviour is that this query returns no results. However, if there are multiple class assignments for a subject, then those are listed in the result set. An example could be XGBoost which is the name of an ML approach as well as the name of an ML library. The query will return a row, if erroneously the same IRI had been used for both concepts, ML approach and ML class.

The quality checks are run regularly. When modelling flaws are detected they are fixed. We are not aware of other ontologies that include quality checks.

3.8. Reasoning

A lightweight approach for modelling ML Ontology has been chosen deliberately. However, some use cases may require sophisticated reasoning [11]. For this reason, a lightweight yet powerful reasoning mechanism is employed in ML Ontology, which has been proposed in own previous work [10].

Consider the following example depicted in Figure 4. For ML tasks, various hierarchies have been specified. For example, classification may be refined according to the number of classes (binary classification, multiclass classification) or according to the dataset types (tabular classification, text classification, etc.). `skos:broader` has been chosen deliberately as a predicate with rather weak semantics to allow for various kinds of hierarchical groupings. The root of this hierarchy, the ML task classification, is specified as belonging to the ML area supervised learning. Logically speaking, one can state that if an ML task belongs to some ML area then all specific variants of this ML task belong to the same ML area, e.g., text classification belongs to supervised learning since classification belongs to supervised learning. In order to foster completeness of ML Ontology, one could add a quality check, ensuring that for all ML tasks, an ML area is assigned which it belongs to, forcing knowledge engineers to manually model all belongs to relationships. However, this is not only tedious but also introduces redundancy that should be avoided.

Another approach, which has been chosen, is to introduce use-case-specific reasoning to add correct `belongs_to` relationships even when they have not been modelled explicitly. Consider the following declaration of the property `belongs_to`.

```
:belongs_to a rdf:Property;
    :inheritance :broader_inherited.
```

Like all properties used as predicates in ML Ontology, `belongs_to` is declared as an RDF Property. In addition, the meta-property `inheritance` specifies its inheritance behaviour as broader inherited. Inheritance is a mechanism adopted in computer science, in particular in object-oriented programming, for propagating properties across hierarchical structures, enabling derived entities to access characteristics of their predecessors. The following use-case-specific rule implements such inheritance behaviour for properties that have been declared as broader inherited.

```
INSERT {?s ?p ?o.}
WHERE  {?p a rdf:Property;
```

 doi:10.20944/preprints202511.1420.v1


```
  :inheritance :broader_inherited.
?s skos:broader+ ?b.
?b ?p ?o.}
```

An `INSERT` statement from the SPARQL Update specification is used which allows inserting new triples into an ontology. This allows implementing production rules that can be used for forward chaining reasoning. In this rule, triples are selected of which predicate `?p` is broader inherited. Then, triples `?b ?p ?o` are selected, for which the subject `?b` is in a direct or indirect broader relationship with an individual `?s`. If those conditions are satisfied then a new triple `?s ?p ?o` is inserted into the ontology.

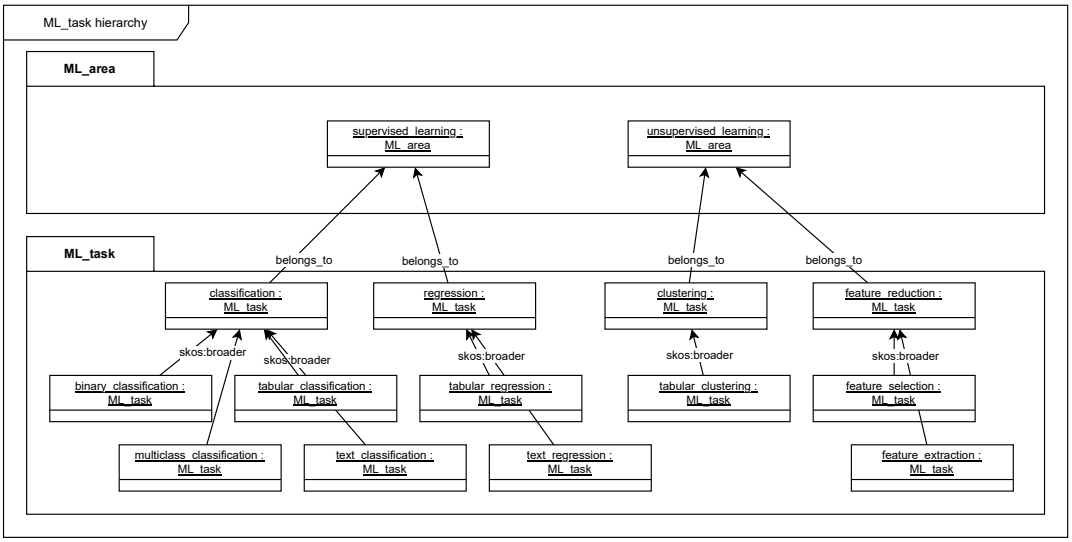

Figure 4. ML task hierarchy

For example, if the query variable `?p` is bound to the predicate belongs to and `?b` is bound to classification and `?s` is bound to text classification, then the following RDF triple is added to the ontology.

```
:text_classification :belongs_to :supervised_learning .
```

This inference rule is stored as part of ML Ontology using the SPIN class `sp:InsertData`. If reasoning is needed in some use case context then all inference rules need to be executed after loading ML Ontology into an RDF store and before executing queries. After this, a query for ML tasks that belongs to supervised learning will also return text classification even if this is not explicitly specified in the ontology.

Note that other predicates used in ML task (e.g., `skos:prefLabel`, `has_dataset_type`, etc.) do not have the meta-property broader inherited and therefore those values are not inherited along the broader hierarchy. So, the label for classification will not be inherited to text classification.

This mechanism allows use-case-specific reasoning to be implemented transparently and easily. Ontology modelling is still lightweight while the full expressive power of SPARQL may be used for specifying complex inference rules.

4. Use Cases

4.1. ML Training Configuration Wizard

One of the primary applications of ontologies is the alignment of heterogeneous vocabularies within a given application domain. Here, a use case involving the configuration wizard of the data science platform OMA-ML [12,13] is presented. OMA-ML operationalises the concept of Meta-AutoML [14]. Meta-AutoML enables the automated generation of ML models for specific datasets by orchestrating multiple competing AutoML [15] implementations. While business domain

 doi:10.20944/preprints202511.1420.v1

experts (e.g., biologists) may want to provide minimal input to configure the training process, AI specialists can fine-tune the underlying parameters to optimise model performance. However, the configuration interfaces and terminologies used across different AutoML solutions vary significantly, posing challenges for interoperability and consistent user experience. In the example of tabular regression, Auto-Sklearn offers 4 parameters for configuring a metric (r2, mean squared error, mean absolute error, median absolute error) while Autokeras offers 7 different parameters (additionally including root mean squared error, mean absolute percentage error, mean squared log error, cosine similarity, log cosh error, but lacking r2 and median absolute error). Also, parameter naming is different between both libraries. What is called `mean_squared_error` in Auto-sklearn, is called `MeanSquaredError` in Autokeras.

An ontology is perfect for aligning the differently used terminologies. ML Ontology is used as the information backbone of the OMA-ML platform. The various configuration options of different AutoML solutions are specified using the classes `Configuration_item`, `ML_library` and `AutoML_solution` as explained in Section 3.5.

The ML training configuration wizard of OMA-ML enables users to set up ML training processes by selecting applicable configuration options provided by the chosen AutoML solutions. Refer to Figure 5 for a screenshot illustrating this functionality. The interface displays configuration parameters such as metric, loss function, max_trials, and tuner for AutoKeras. For the tuner parameter, selectable options are presented in a dropdown menu, including greedy, bayesian, random, and hyperband. All parameters and their corresponding values are dynamically retrieved from ML Ontology.

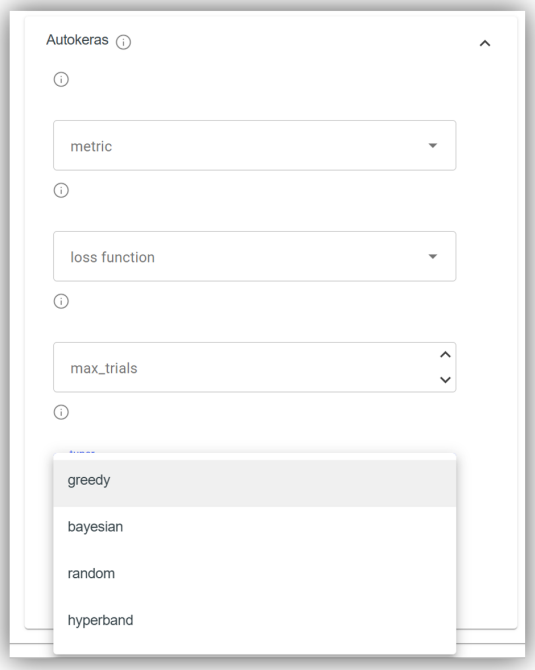

Figure 5. Use case configuration wizard

4.2. Interactive Help

Another use case is interactive help. Data science platforms such as OMA-ML are used by persons with varying levels of expertise in AI and ML. To support this diversity, it is beneficial to provide accessible information not only on core ML concepts—such as ML areas, ML tasks, ML approaches, and metrics—but also on implementations, including ML libraries and AutoML solutions. The alignment of ML Ontology with Wikidata plays a crucial role in enabling this functionality. Approximately 350 ML-related terms are linked to corresponding Wikidata entries, which in turn connect to Wikipedia articles available in multiple languages, facilitating multilingual and context-aware assistance. See Figure 6 for a screenshot illustrating the selection interface for ML libraries and AutoML solutions in

 doi:10.20944/preprints202511.1420.v1

OMA-ML. When hovering over a library (e.g., Scikit-learn), a pop-up appears displaying an image and summary text from the English Wikipedia article, along with a link for further reading.

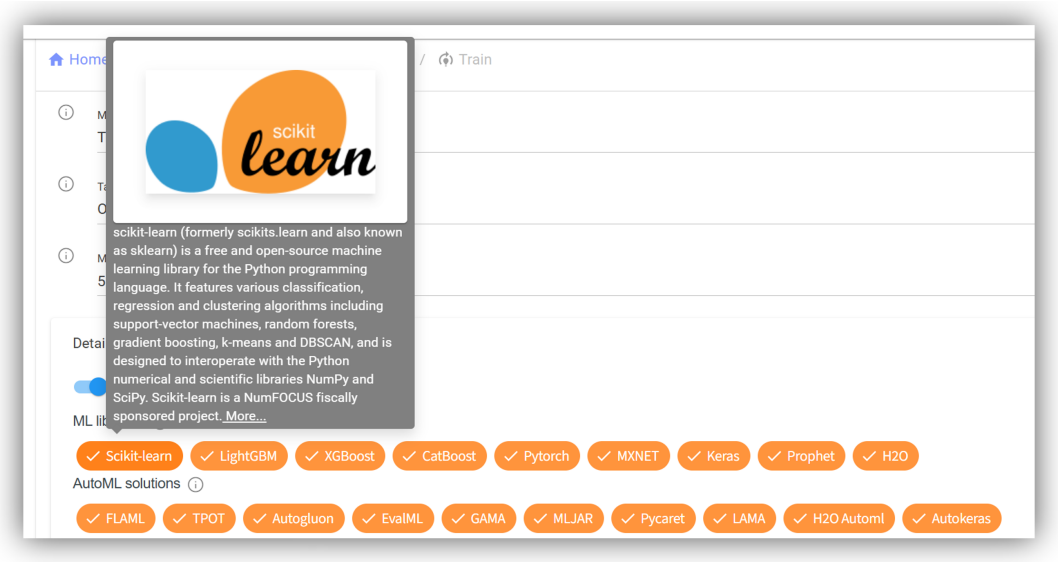

Figure 6. Use case interactive help

5. Discussion

In this section, the ML Ontology architecture is compared with requirements specified in Section 1.

1. **Volume**: ML Ontology is notably comprehensive, encompassing ca. 700 individuals that define key machine learning concepts. With roughly 5,000 RDF triples, it ranks among the largest domain-specific ontologies for ML that we are aware of. The only ontology for the ML domain comparable in volume is OntoDM with ca 660 classes representing ML concepts, described by ca. 3,700 annotations (AnnotationAssertions, subClassOf relationships, AnnotationProperties, DisjointClasses etc.). The ML Schema Core Specification as an upper-level ontology only specifies 25 classes. According to [4], DMOP includes ca. 720 classes representing ML concepts, described by ca. 4,300 annotations (data properties, logical axioms etc.). However, the ontology does not seem to be publicly available, just like the Exposé ontology for data mining experiments. Finally, the MEX vocabulary comprises ca. 250 classes.
2. **Performance**: ML Ontology allows high-performance query access. A benchmark has been performed to execute SPARQL queries joining several classes with 100+ results (Setup: In-memory SPARQL engine: rdflib 7.1.4, CPU: Intel Core Ultra 7 165H, 3.80 GHz, 32 GB RAM). On average, a SPARQL query executed in 2.9 ms. We consider this fast enough for industry applications.
3. **Balance between simplicity and expressiveness**: ML Ontology deliberately uses lightweight modelling approach (knowledge graph implemented in RDF/RDFS) which is particularly suited for industry use. At the same time, ML ontology offers the full expressive power of SPARQL inference rules that can be used in a use-case-specific manner.
4. **Extensibility and adaptability**: ML Ontology is modularised and can be easily extended and adapted to various use cases. The ontology schema provides a clear separation between classes, instances, inference rules, queries and QA checks (see Figure 1). ML Ontology classes are separated in ML concepts and ML implementations (see Figure 2). ML Ontology has been extended continuously over a period of more than five years and the ontology schema has proven stable.
5. **Built-in quality management**: ML Ontology comprises built-in quality checks that can be adapted use-case-specifically in order to ensure that quality requirements from the industry are met. As opposed to just documenting modelling guidelines, quality checks can be executed regularly,

 doi:10.20944/preprints202511.1420.v1

allowing guideline violations to be detected and fixed. ML Ontology has be co-edited by more than 15 knowledge engineers over more than 5 years and still has not deteriorated in quality.

6. **Standards**: ML Ontology is solely based on Semantic Web (SW) standards including RDF/RDFS and SPARQL, and uses state-of-the-art ontology schemas such as SKOS and SPIN. Experiments have shown that ML Ontology can easily be deployed as a Labeled Property Graph (LPG) if needed. ML Ontology could be imported without adaptation into Neo4J and the pre-defined SPARQL queries could be automatically converted to correct Cypher queries.
7. **Public availability**: ML Ontology is published open source[26] under an MIT license.

In total, the criteria sufficient volume, performance, expressiveness, extensibility and adaptability, quality and conformance to standards are fulfilled. Therefore, ML ontology can be regarded as industry-ready.

6. Conclusions and Future Work

In this article, ML Ontology has been presented, an industry-ready ontology for the ML domain. ML Ontology is comprehensive, provides good performance and is extensible and adaptable. In comparison to existing work, ML Ontology provides novel features, including built-in queries and quality assurance. Notably, ML Ontology provides sophisticated reasoning while being based on lightweight modelling languages (RDF/RDFS). Its industry-readiness has been demonstrated by benchmarks and use case implementations within a data science platform.

ML Ontology can be considered ready for use in industry applications requiring the formalisation of ML concepts and their relationships. As future work, we envisage the extension of ML Ontology as an incremental, use-case-driven process. Further applications accessing ML Ontology may define requirements for extensions, e.g., the parametrisation of ML libraries. As ML Ontology is open source, we regard this as a community effort.

References

1. Gruber, T.R. A translation approach to portable ontology specifications. *Knowledge acquisition* **1993**, *5*, 199–220.
2. Publio, G.C.; Esteves, D.; Panov, P.; Soldatova, L.; Soru, T.; Vanschoren, J.; Zafar, H.; et al. ML-schema: exposing the semantics of machine learning with schemas and ontologies. *arXiv preprint arXiv:1807.05351* **2018**.
3. Panov, P.; Soldatova, L.; Džeroski, S. Ontology of core data mining entities. *Data Mining and Knowledge Discovery* **2014**, *28*, 1222–1265.
4. Keet, C.M.; Ławrynowicz, A.; d'Amato, C.; Kalousis, A.; Nguyen, P.; Palma, R.; Stevens, R.; Hilario, M. The data mining optimization ontology. *Journal of web semantics* **2015**, *32*, 43–53.
5. Vanschoren, J.; Soldatova, L. Exposé: An ontology for data mining experiments. In Proceedings of the International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010), 2010, pp. 31–46.
6. Anikin, D.; Borisenko, O.; Nedumov, Y. Labeled property graphs: SQL or NoSQL? In Proceedings of the 2019 Ivannikov Memorial Workshop (IVMEM). IEEE, 2019, pp. 7–13.
7. Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; Melo, G.D.; Gutierrez, C.; Kirrane, S.; Gayo, J.E.L.; Navigli, R.; Neumaier, S.; et al. Knowledge graphs. *ACM Computing Surveys (Csur)* **2021**, *54*, 1–37.
8. Humm, B.G.; Archer, P.; Bense, H.; Bernier, C.; Goetz, C.; Hoppe, T.; Schumann, F.; Siegel, M.; Wenning, R.; Zender, A. New directions for applied knowledge-based AI and machine learning: Selected results of the 2022 Dagstuhl Workshop on Applied Machine Intelligence. *Informatik Spektrum* **2023**, *46*, 65–78.
9. Kifer, M.; Lausen, G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In Proceedings of the Proceedings of the 1989 ACM SIGMOD international conference on Management of data, 1989, pp. 134–146.
10. Bense, H.; Humm, B.G. An Extensible Approach to Multi-level Ontology Modelling. In Proceedings of the KMIS, 2021, pp. 184–193.
11. Huth, M.; Ryan, M. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2 ed.; Cambridge University Press, 2004.

[26] https://github.com/hochschule-darmstadt/MetaAutoML/blob/main/controller/managers/ontology/ML_Ontology.ttl

 doi:10.20944/preprints202511.1420.v1

12. Zender, A.; Humm, B.G. Ontology-based meta automl. *Integrated Computer-Aided Engineering* **2022**, *29*, 351–366.
13. Zender, A.; Humm, B.G.; Holzheuser, A. Enhancing User Experience in Artificial Intelligence Systems: A Practical Approach. In Proceedings of the Topical Area: Software, System and Service Engineering (S3E) of the FedCSIS Conference on Computer Science and Intelligence Systems. Springer, 2024, pp. 113–131.
14. Humm, B.G.; Zender, A. An ontology-based concept for meta automl. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, 2021, pp. 117–128.
15. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2015.