# ML Schema Core Specification

## Release 17 October 2016

**Latest version**

http://www.w3.org/2016/10/mls/

**Editors:**

Agnieszka Ławrynowicz, Poznan University of Technology
Panče Panov, Jozef Stefan Institute
Larisa Soldatova, Brunel University
Tommaso Soru, University of Leipzig
Joaquin Vanschoren, Eindhoven University of Technology

(alphabetic order)

**Authors:**

Diego Esteves, University of Leipzig
Agnieszka Ławrynowicz, Poznan University of Technology
Panče Panov, Jozef Stefan Institute
Larisa Soldatova, Brunel University
Tommaso Soru, University of Leipzig

Joaquin Vanschoren, Eindhoven University of Technology
(alphabetic order)



---

## Abstract

The ML Schema is a simple shared schema that provides a set of classes, properties, and restrictions that can be used to represent and interchange information on data mining and machine learning algorithms, datasets, and experiments. It can be specialized to create new classes and properties. It can be mapped to more complex, specific ontologies on data mining and machine learning, and also used as a basis for markup languages and data exchange standards.

The namespace for ML Schema terms is http://www.w3.org/ns/mls#.

The OWL encoding of the ML Schema is available here.

## Status of This Document

## Table of Contents



## 1. Introduction

The core vocabulary of ML Schema deals with machine learning (ML) algorithms. The schema can be used to represent the algorithms, the machine learning tasks they address, their implementations and executions, as well as inputs (e.g., data) and outputs (e.g., models) they specify.

This lightweight schema may be used as a basis for ontology development projects, markup languages and data exchange standards. In particular, it aims to align existing machine learning ontologies and to support development of more specific ontologies with specific purposes/applications. The main purpose is to increase interoperability by preventing a proliferation of incompatible machine learning ontologies as well as to provide a high-level standard to represent machine learning data. Thus, this scenario leads to a more representative and comprehensive ontology derived from existing state-of-the-art ML schemas.

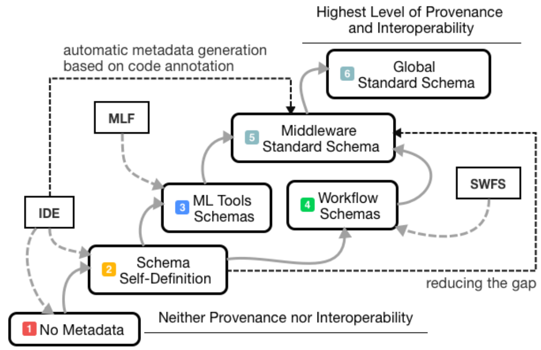
The schema also defines a relationship between machine learning algorithms and their single executions and experiments and studies encompassing them. It aims at stimulating the development of standards in order to achieve high level of interoperability among scientific experiments concerning machine learning. By facilitating the metadata interchange process, the ML Schema may foster reproducible research. Another goal of ML Schema related to interoperability and reproducible research it to facilitate turning machine learning algorithms and results into linked open data.

## 1.1 Benefits

Despite recent efforts to achieve a high level of interoperability of Machine Learning (ML) experiments we still run into problems created due to the existence of different ML platforms: each of those have a specific conceptualization or schema for representing data and metadata (Fig1: itens 3 and 4: **vertical interoperability**). This scenario leads to an extra coding-effort (Fig1: item 2) to achieve both the desired interoperability and a better provenance level as well as a more automatized environment for obtaining the generated results. To reduce the gap, ML vocabularies and ontologies have been proposed (Figure 1: item 5).

- Onto-DM has been designed to provide generic representations of principle entities in the area of data mining.
- DMOP has been developed with a purpose to support meta-mining, i.e. meta-learning from the full ML process.
- Expose is designed to describe (and reason about) machine learning experiments. It is built on top of OntoDM, and underlies OpenML, a collaboration and meta-learning platform for machine learning.
- MEX Vocabulary (mex-core, mex-algo and mex-perf) has been designed to tackle the problem of managing machine learning outcomes and sharing provenance information particularly on the basic machine learning iterations in a lightweight format.

The gap can be further significantly reduced by achieving interoperability among state-of-the-art (SOTA) schemata of those resources (Figure 1: item 5) i.e. achieving the **horizontal interoperability** (Figure 1: item 6). Therefore, different groups of researchers could exchange SOTA metadata files in a transparent manner, e.g.: from OntoDM and MEX (`MLSchema.Schema data = mlschema.convert('myfile.ttl', MLSchema.Ontology.OntoDM, MLSchema.Ontology.MEX)`).




[Figure 1](#) - Vertical and Horizontal Interoperability across ML Environments.

Besides a higher level of interoperability, it direct benefits ML Ecosystems (e.g.: [OpenML](#)) and ML Metadata Repositories (e.g.: [WASOTA](#)) which can rely on a more representative standard in their architecures.

In [OpenML](#), MLSchema is used to export all machine learning datasets, tasks, workflows and runs as linked open data (in RDF). This allows scientists to connect the results of their machine learning experiments to other knowledge sources, or to build novel knowledge bases for machine learning research.

## 1.2 Notational Conventions

Throughout this document, we use [Turtle RDF Syntax](#) to express examples showing the use of the schema.

## 1.3 Namespaces

*This section is non-normative.*

The following namespace prefixes are used throughout this document.

Table 1: Prefix and Namespaces used in this specification

| prefix | namespace IRI | definition |
|---|---|---|
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# | The RDF namespace [RDF-CONCEPTS] |
| xsd | http://www.w3.org/2000/10/XMLSchema# | XML Schema Namespace [XMLSCHEMA11-2] |
| owl | http://www.w3.org/2002/07/owl# | The OWL namespace [OWL2-OVERVIEW] |
| mls | http://www.w3.org/ns/mls# | The ML Schema namespace |
| frapo | http://www.sparontologies.net/ontologies/frapo; http://purl.org/cerif/frapo | Funding, Research Administration and Projects Ontology |
| dc | http://purl.org/dc/elements/1.1/ | Dublin Core |

| | | |
|---|---|---|
| | | [DUBLIN-CORE] |
| foaf | http://xmlns.com/foaf/0.1/ | FOAF Vocabulary Specification [FOAF] |
| (others) | (various) | Other namespace prefixes appearing only in examples. |

## 2. The ML Schema description

The following diagram depicts the core ML Schema. Boxes represent classes of the schema. Arrows without filled heads represent properties, arrows with empty heads represent subclass relations, and arrows with diamonds represent part-of relations.

**IRI:**
http://www.w3.org/ns/mls#
**Version IRI:**
http://www.w3.org/2016/10/mls#

**Other visualisation:**
Ontology source

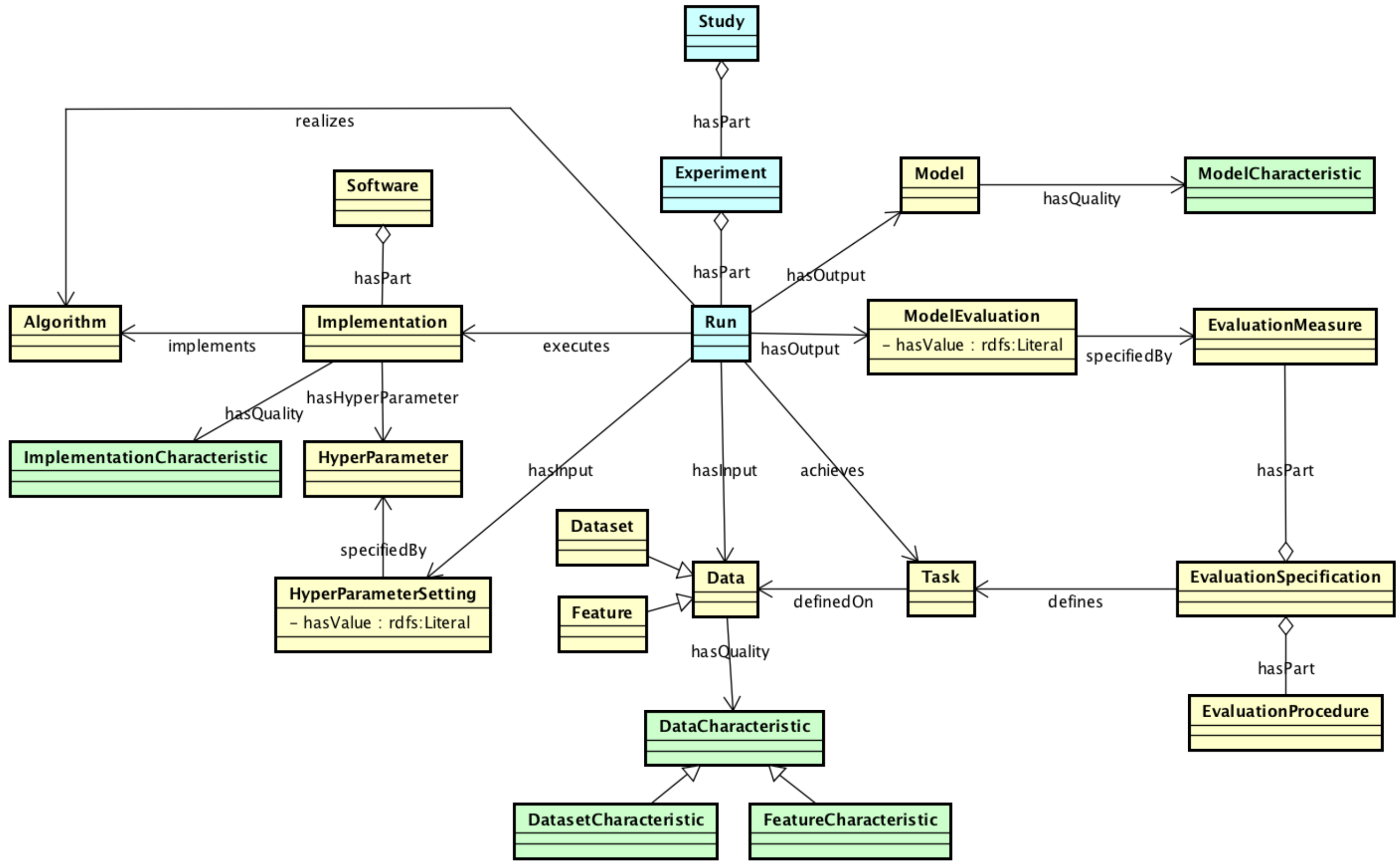



Figure 2. The ML Schema core vocabulary.
The diagram depicts Information Entities as yellow boxes, Processes as blue boxes, and Qualities as green boxes.

We will illustrate the ML schema by means of two examples.

Firstly, we illustrate the ML schema with an example derived from the OpenML portal (see Fig. 3). The example describes entities involved to model a single run of the implementation of a logistic regression algorithm from a Weka machine learning environment. The referenced individuals can easily be looked up online. For instance, run 100241 can be found on [http://www.openml.org/r/100241](http://www.openml.org/r/100241).

### EXAMPLE 1

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

:run100241 rdf:type owl:NamedIndividual ,
                    mls:Run ;

           mls:executes :wekaLogistic ;

           mls:hasInput :credit-a ,
                        :wekaLogisticMSetting29 ,
                        :wekaLogisticRSetting29 ;

           mls:hasOutput :modelEvaluation100241 ,
                         :wekaLogisticModel100241 ;

           mls:realizes :logisticRegression ;

           mls:achieves :task29 .


:wekaLogistic rdf:type owl:NamedIndividual ,
                       mls:Implementation ;

              mls:hasHyperParameter :wekaLogisticC ,
                                    :wekaLogisticDoNotCheckCapabilit
                                    :wekaLogisticM ,
```

```
                                                :wekaLogisticOutputDebugInfo ,
                                                :wekaLogisticR ;

                   mls:implements :logisticRegression .

:weka rdf:type mls:Software,
                   mls:hasPart :wekaLogistic.


:logisticRegression rdf:type owl:NamedIndividual ,
                                        mls:Algorithm .

:wekaLogisticC rdf:type owl:NamedIndividual ,
                                 mls:HyperParameter .

:wekaLogisticDoNotCheckCapabilities rdf:type owl:NamedIndividual ,
                                                      mls:HyperParameter .

:wekaLogisticM rdf:type owl:NamedIndividual ,
                                 mls:HyperParameter .

:wekaLogisticOutputDebugInfo rdf:type owl:NamedIndividual ,
                                               mls:HyperParameter .

:wekaLogisticR rdf:type owl:NamedIndividual ,
                                 mls:HyperParameter .

:wekaLogisticMSetting29 rdf:type owl:NamedIndividual ,
                                        mls:HyperParameterSetting ;

                                 mls:specifiedBy :wekaLogisticM ;

                                 mls:hasValue -1 .

:wekaLogisticRSetting29 rdf:type owl:NamedIndividual ,
                                        mls:HyperParameterSetting ;
```

```
                        mls:specifiedBy :wekaLogisticR ;

                        mls:hasValue "1.0E-8"^^xsd:float .


:credit-a rdf:type owl:NamedIndividual ,
                 mls:Dataset ;

          mls:hasQuality :defaultAccuracy ,
                         :numberOfFeatures ,
                         :numberOfInstances .


:defaultAccuracy rdf:type owl:NamedIndividual ,
                          mls:DatasetCharacteristic ;
                 mls:hasValue "0.56"^^xsd:float .

:numberOfFeatures rdf:type owl:NamedIndividual ,
                                mls:DatasetCharacteristic ;
                 mls:hasValue "16"^^xsd:long .

:numberOfInstances rdf:type owl:NamedIndividual ,
                            mls:DatasetCharacteristic ;
                 mls:hasValue "690"^^xsd:long .


:wekaLogisticModel100241 rdf:type owl:NamedIndividual ,
                                  mls:Model .

:modelEvaluation100241 rdf:type owl:NamedIndividual ,
                                mls:ModelEvaluation ;

                       mls:specifiedBy :predictiveAccuracy ;

                       mls:hasValue 0.8478 .

:predictiveAccuracy rdf:type owl:NamedIndividual ,
```

```
                                 mls:EvaluationMeasure .

:task29 rdf:type owl:NamedIndividual ,
                 mls:Task ;

        mls:definedOn :credit-a .


:evaluationSpecification1 rdf:type owl:NamedIndividual ,
                                   mls:EvaluationSpecification ;

                          mls:defines :task29 ;

                          mls:hasPart :TenFoldCrossValidation ,
                                      :predictiveAccuracy .

:TenFoldCrossValidation rdf:type owl:NamedIndividual ,
                                 mls:EvaluationProcedure .
```

In the example, the run `:run100241` executes the implementation `:wekaLogistic` of the algorithm `:logisticRegression` which this execution realizes. The run has on input the `:credit-a` dataset and its output is both the model `:wekaLogisticModel100241` and the model evaluation `:modelEvaluation100241`. The run achieves the task `:task29`.

The implementation `:wekaLogistic` implements the algorithm `:logisticRegression`. The implementation has five hyperparameters: `:wekaLogisticC`, `:wekaLogisticDoNotCheckCapabilities`, `:wekaLogisticM`, `:wekaLogisticOutputDebugInfo`, `:wekaLogisticR`. Two of these hyperparameters are set for the run `:run100241`. The hyperparameter `:wekaLogisticM` has value set to `-1` (expressed via the hyperparameter setting `:wekaLogisticMSetting29`), and the hyperparameter `:wekaLogisticR` that has its value set to `"1.0E-8"^^xsd:float` (expressed via the hyperparameter setting `:wekaLogisticRSetting29`).

The dataset `:credit-a` has several characteristics such as: `:decisionStumpAUC`, `:defaultAccuracy`, `:numberOfInstances`, `:numberOfMissingValues`.

The predictive model `:wekaLogisticModel100241` is evaluated (`:modelEvaluation100241`) based on the specified evaluation measure `:predictiveAccuracy`. The value of the evaluation measure modeled via the model evaluation `:modelEvaluation100241` is `0.8478`.

The task `:task29` is defined on the dataset (`credit-a`) and on the evaluation specification `:evaluationSpecification1`. The evaluation specification `:evaluationSpecification1` has parts: the evaluation procedure `:TenFoldCrossValidation` and the evaluation measure`:predictiveAccuracy`.

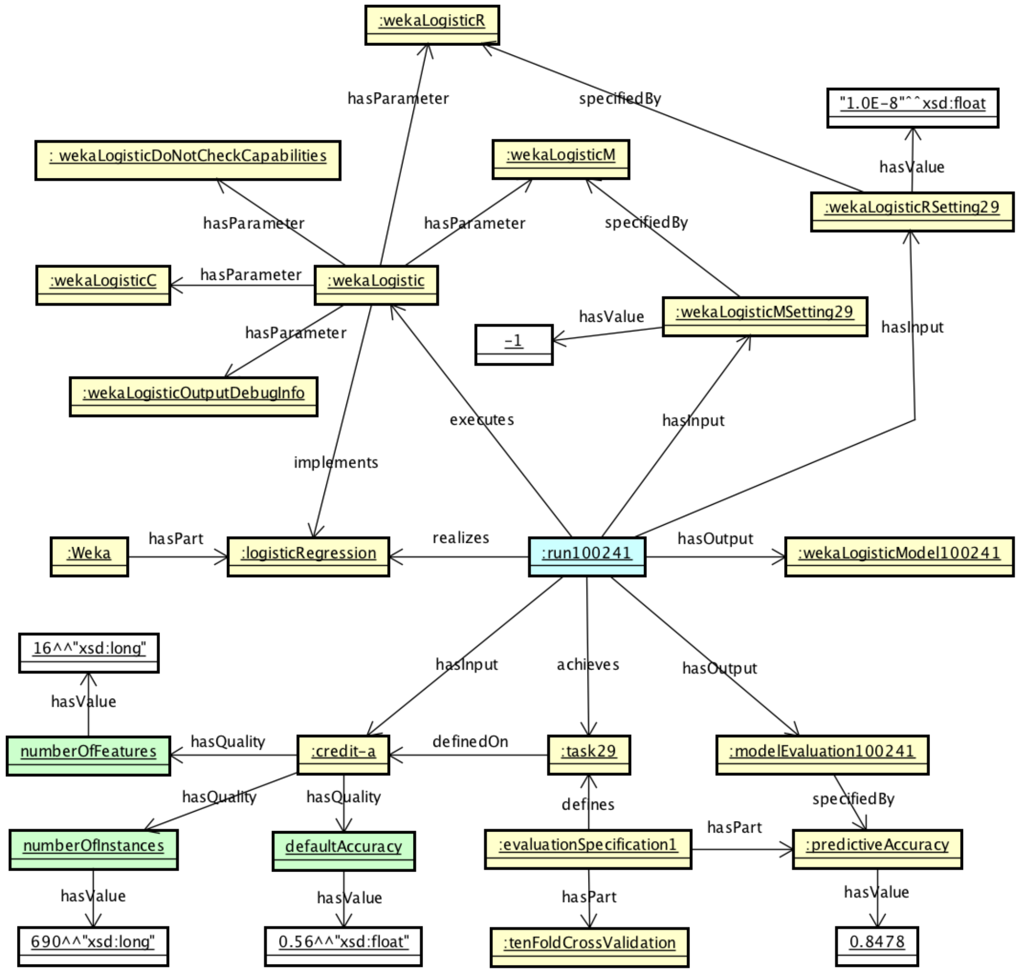



Figure 3. An example illustrating a single run of an ML algorithm implementation.
The diagram depicts Information Entities as yellow boxes, Processes as blue boxes, and Material Entities as red boxes.

Secondly, we illustrate ML schema with an example describing ML study (`:study1`) and the corresponding dataset `:mtl_dataset`, providing reference to a publication (`:article1`), and acknowledging the funding body (see Fig. 3) . This example refers to the article

“Multi-Task Learning with a Natural Metric for Quantitative Structure Activity Relationship Learning” by Sadawi et al which reports on the ML study carried out within the Meta-QSAR project (`:meta-qsar_project`) funded by `:EPSRC` (`:grant1` with number EP/K030582/1). The referred dataset is freely available in OpenML.

Exposing such metadata may be of use for possible collaborators who may wish to analyse research networks and try to assess the 'trustwothiness' of what is published in the literature. Such information that a study is done within a funded project, may increase their level of trust to the published results.

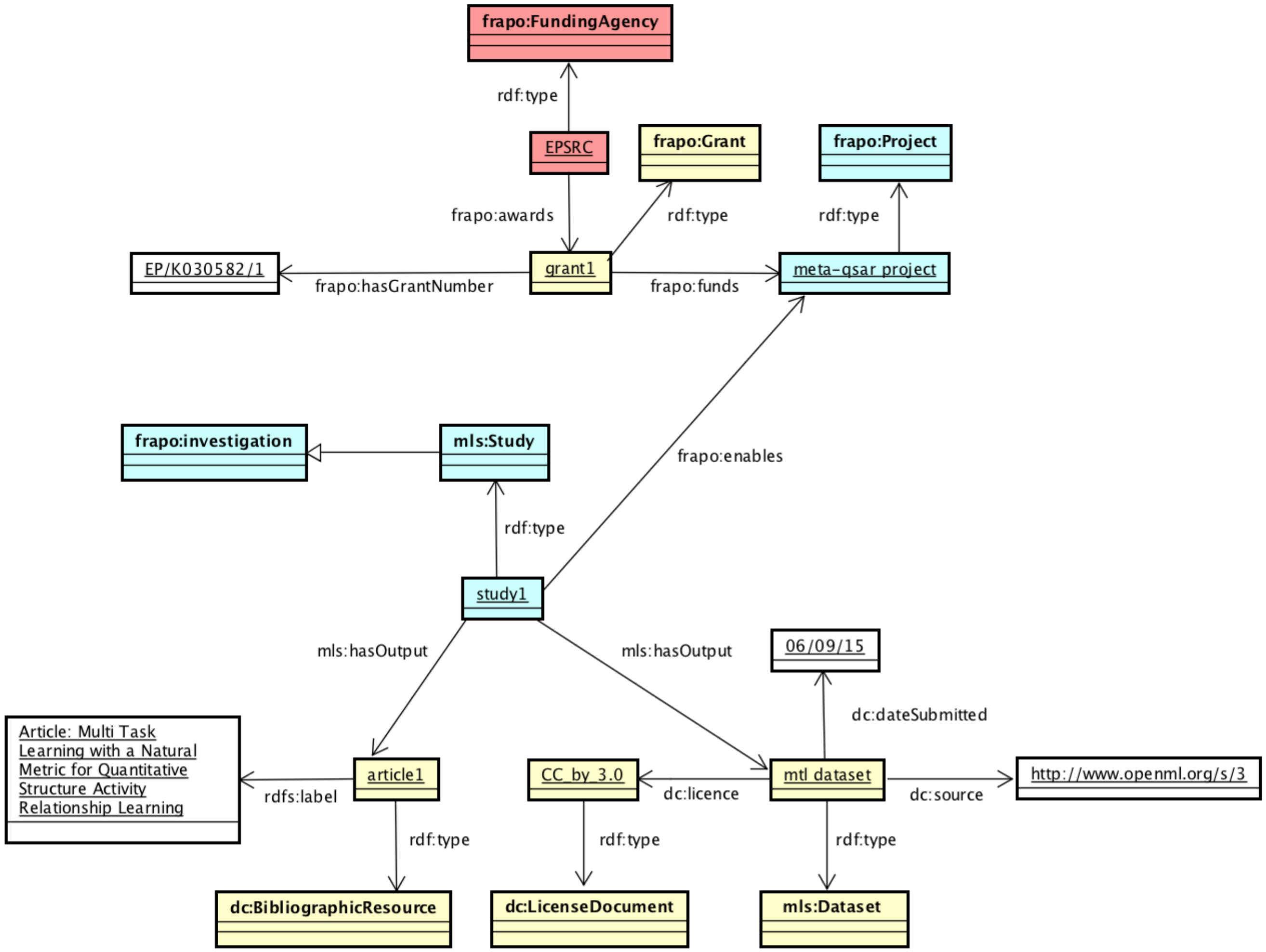



Figure 4. An example illustrating ML study.

## EXAMPLE 2

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix dc:  <http://purl.org/dc/elements/1.1/>> .
```

```
@prefix frapo: <http://purl.org/cerif/frapo/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .


dc:BibliographicResource rdf:type owl:Class .


mls:Study rdfs:subClassOf frapo:Investigation .

:EPSRC rdf:type frapo:FundingAgency ,
                owl:NamedIndividual ;

       frapo:awards :grant1 .

:article1 rdf:type dc:BibliographicResource ,
                   owl:NamedIndividual ;

          rdfs:label "Article: Multi Task Learning with a Natural Met

:grant1 rdf:type frapo:Grant ,
                 owl:NamedIndividual ;

        frapo:hasGrantNumber "EP/K030582/1" ;

        frapo:funds :meta-qsar_project .


:meta-qsar_project rdf:type owl:NamedIndividual ,
                            foaf:Project .

:mtl_dataset rdf:type owl:NamedIndividual ,
                      mls:Dataset ;

             dc:licence :CC_by_3.0 ;
             dc:dateSubmitted "06/09/15" ;
             dc:source "http://www.openml.org/s/3" .
```

```
:CC_by_3.0 rdf:type owl:NamedIndividual ,
                    dc:LicenceDocument .

:study1 rdf:type owl:NamedIndividual ,
                mls:Study ;

        frapo:enables :meta-qsar_project ;

        frapo:hasOutput :mtl_dataset .
```

# Classes

- algorithm
- data
- data characteristic
- dataset
- dataset characteristic
- evaluation measure
- evaluation procedure
- evaluation specification
- experiment
- feature
- feature characteristic
- hyper parameter
- hyper parameter setting
- implementation
- implementation characteristic
- information entity

## algorithm[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Algorithm

The algorithm regardless software implementation.

**has super-classes**

information entity[c]

EXAMPLE 3

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:logisticRegression rdf:type owl:NamedIndividual ,
                             mls:Algorithm .
```

## data[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Data

Data is a data item composed of data examples and it may be of a various level of granularity and complexity. With regard to granularity, it can be a whole dataset (for instance, one main table and possibly other tables), or only a single table, or only a feature (e.g., a column of a table), or only an instance (e.g., row of a table), or a single feature-value pair. With regard to complexity, data examples are characterized by their datatype, which may be arbitrarily complex (e.g., instead of a table it can be an arbitrary graph).

**has super-classes**

information entity[c]

has quality [op] some data characteristic[c]

**has sub-classes**

dataset[c], feature[c]

## data characteristic[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#DataCharacteristic

DataCharacteristic is a distinguishing quality or property that distinguish one data from another. Such characteristics are often statistical ones (e.g., the number of instances or the number of features of a data set). They may be also informationtheoretic measures (e.g., class entropy of a categorical data set) or geometric measures of data complexity (e.g., the highest discriminatory power of any single feature in the data set).

**has super-classes**

information entity[c]

**has sub-classes**

dataset characteristic[c], feature characteristic[c]

## dataset[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Dataset

**has super-classes**

data[c]

EXAMPLE 4

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:credit-a rdf:type owl:NamedIndividual ,
                   mls:Dataset ;

          mls:hasQuality :decisionStumpAUC ,
                         :defaultAccuracy ,
                         :numberOfInstances ,
                         :numberOfMissingValues .
```

## dataset characteristic[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#DatasetCharacteristic

**has super-classes**

data characteristic[c]

EXAMPLE 5

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:numberOfInstances rdf:type owl:NamedIndividual ,
                            mls:DatasetCharacteristic .

:numberOfMissingValues rdf:type owl:NamedIndividual ,
                                mls:DatasetCharacteristic .

:decisionStumpAUC rdf:type owl:NamedIndividual ,
                           mls:DatasetCharacteristic .

:defaultAccuracy rdf:type owl:NamedIndividual ,
                          mls:DatasetCharacteristic .
```

## evaluation measure[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#EvaluationMeasure

EvaluationMeasure is a measure to assess the performance of the model generated by the process that realizes the task. Examples are predictive accuracy or f-measure.

**has super-classes**

information entity[c]

## evaluation procedure[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#EvaluationProcedure

EvaluationProcedure is a technique to evaluate machine learning models. Examples are cross-validation and leave-one-out.

**has super-classes**

information entity[c]

EXAMPLE 6

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:TenFoldCrossValidation rdf:type owl:NamedIndividual ,
                                 mls:EvaluationProcedure .
```

## evaluation specification[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#EvaluationSpecification

EvaluationProcedure is a technique to evaluate machine learning models. Examples are cross-validation and leave-one-out.

### has super-classes

information entity[c]
hasPart[op] some evaluation procedure[c]
hasPart[op] some evaluation measure[c]

EXAMPLE 7

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:evaluationSpecification1 rdf:type owl:NamedIndividual ,
                                   mls:EvaluationSpecification ;

                          mls:defines :task29 ;

                          mls:hasPart :TenFoldCrossValidation ,
                                      :predictiveAccuracy .
```

## experiment[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Experiment

Experiment is a collection of runs.

**has super-classes**

Process[c]
hasPart[op] some run[c]

## feature[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Feature

**has super-classes**

data[c]

## feature characteristic[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#FeatureCharacteristic

**has super-classes**

data characteristic[c]

## hyper parameter[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#HyperParameter

Hyperparameter is a prior parameter of an implementation, i.e., a parameter which is set before its execution (e.g. C, the complexity parameter, in weka.SMO).

**has super-classes**

information entity[c]

EXAMPLE 8

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:wekaLogisticC rdf:type owl:NamedIndividual ,
                        mls:HyperParameter .

:wekaLogisticDoNotCheckCapabilities rdf:type owl:NamedIndividual ,
                                             mls:HyperParameter .

:wekaLogisticM rdf:type owl:NamedIndividual ,
                        mls:HyperParameter .

:wekaLogisticOutputDebugInfo rdf:type owl:NamedIndividual ,
                                      mls:HyperParameter .

:wekaLogisticR rdf:type owl:NamedIndividual ,
                        mls:HyperParameter .
```

## hyper parameter setting[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#HyperParameterSetting

HyperParameterSetting is an entity which connects a hyperparameter and its value that is being set before an implementation execution.

### has super-classes

information entity[c]
specified by[op] some hyper parameter[c]
has value[dp] some literal

EXAMPLE 9

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .


:wekaLogisticMSetting29 rdf:type owl:NamedIndividual ,
                                 mls:HyperParameterSetting ;

                        mls:specifiedBy :wekaLogisticM ;

                        mls:hasValue -1 .

:wekaLogisticRSetting29 rdf:type owl:NamedIndividual ,
                                 mls:HyperParameterSetting ;

                        mls:specifiedBy :wekaLogisticR ;

                        mls:hasValue "1.0E-8"^^xsd:float .
```

## implementation^c back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Implementation

Implementation is an executable implementation of a machine learning algorithm, a script, or a workflow. It is versioned, and sometimes belongs to a library (e.g. WEKA).

### has super-classes

information entity^c
hasHyperParameter^op some hyper parameter^c
has quality^op some implementation characteristic^c
implements^op some algorithm^c

EXAMPLE 10

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:wekaLogistic rdf:type owl:NamedIndividual ,
                       mls:Implementation ;

              mls:hasHyperParameter :wekaLogisticC ,
                                    :wekaLogisticDoNotCheckCapabilit
                                    :wekaLogisticM ,
                                    :wekaLogisticOutputDebugInfo ,
                                    :wekaLogisticR ;
```

```
            mls:implements :logisticRegression .
```

## implementation characteristic^c back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#ImplementationCharacteristic

ImplementationCharacteristic is a distinguishing quality or property that distinguish one implementation from another.

**has super-classes**

information entity^c

## model^c back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Model

Model is a generalization of a set of training data able to predict values for unseen instances. It is an output from an execution of a data mining algorithm implementation. Models have a dual nature. They can be treated as data structures and as such represented, stored and manipulated. On the other hand, they act as functions and are executed, taking as input data examples and giving as output the result of applying the function to a data example. Models can also be divided into global or local ones. A global model has global coverage of a data set, i.e., it generalizes the whole data set. A local model, such as a pattern set, is a set of local hypotheses, i.e. each applies to a limited region of the data set.

### has super-classes

[information entity](#)c
[has quality](#)op some [model characteristic](#)c

EXAMPLE 11

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:wekaLogisticModel100241 rdf:type owl:NamedIndividual ,
                                  mls:Model .
```

## InformationEntityc back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#InformationEntity

### has sub-classes

algorithmc, datac, evaluation measurec, evaluation procedurec, evaluation specificationc, hyper parameterc, hyper parameter settingc, implementationc, modelc, model evaluationc, softwarec, task

## model characteristic[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#ModelCharacteristic

ModelCharacteristic is a distinguishing quality or property that distinguish one model from another. An example model characetristic may be interpretabilty or a complexity of the model.

**has super-classes**

information entity[c]

## model evaluation[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#ModelEvaluation

ModelEvaluation is a setting of a value of the performance measure specified by the evaluation specification. It connects a measure specification with its value.

**has super-classes**

information entity[c]
has value[dp] some literal
specified by[op] some evaluation measure[c]

EXAMPLE 12

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:modelEvaluation100241 rdf:type owl:NamedIndividual ,
                                mls:ModelEvaluation ;

                       mls:specifiedBy :predictiveAccuracy ;

                       mls:hasValue 0.8478 .
```

## Process[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Process

**has sub-classes**

experiment[c], run[c], study[c]

## Quality[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Quality

**has sub-classes**

data characteristic[c], model characteristic[c], implementation characteristic[c]

## run[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Run

Run is an execution of an implementation on a machine (computer). It is limited in time (has a start and end point), can be successful or failed.

**has super-classes**

Process[c]
has output[op] some model[c]
has output[op] some model evaluation[c]
hasInput[op] some data[c]
hasInput[op] some hyper parameter setting[c]
realizes[op] some task[c]
executes[op] some implementation[c]

EXAMPLE 13

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:run100241 rdf:type owl:NamedIndividual ,
                    mls:Run

           mls:executes :wekaLogistic ;

           mls:hasInput :credit-a ,
                        :wekaLogisticMSetting29 ,
                        :wekaLogisticRSetting29 ;

           mls:hasOutput :modelEvaluation100241 ,
                         :wekaLogisticModel100241 ;
```

```
                mls:realizes :task29 .
```

## software[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Software

Software is Implemented computer programs, procedures, scripts or rules with associated documentation, possibly constituting an organized environment, stored in read/write memory for the purpose of being executed within a computer system.

**has super-classes**

information entity[c]
hasPart[op] some implementation[c]

EXAMPLE 14

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


:weka rdf:type mls:Software,
              mls:hasPart :wekaLogistic.
```

## study[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Study

Study is a collection of runs that belong together to do some kind of analysis on its results. This analysis can be general or very specific (e.g. a hypothesis test). Can be linked to files, data, that belong to it.

**has super-classes**

Process[c]
hasPart[op] some experiment[c]

## task[c] back to ToC or Class ToC

**IRI:** http://www.w3.org/ns/mls#Task

Task is a formal description of a process that needs to be completed (e.g. based on inputs and outputs). A Task is any piece of work that needs to be addressed in the data mining process. In ML Schema, it is defined based on data.

**has super-classes**

information entity[c]
definedOn[op] some data[c]

EXAMPLE 15

```
@prefix : <http://example.org#> .
@prefix mls: <http://www.w3.org/ns/mls#> .
```

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:task29 rdf:type owl:NamedIndividual ,
                 mls:Task ;

        mls:definedOn :credit-a .
```

# Object Properties



## achieves^op back to ToC or Object Property ToC

**IRI:** http://www.w3.org/ns/mls#achieves

A relation between a run and a task, where the run achieves specifications formulated by the task..

**has super-properties**
top object property

## definedOn^op back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#definedOn

A relation between a task and either the data or an evaluation specification pertinent to this task.

**has super-properties**
top object property

## defines^op back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#defines

A relation between an evaluation specification and a task.

**has super-properties**
top object property

## executes^op back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#executes

A relation between a run and an implemantation that is being executed during the run.

**has super-properties**
top object property

## hasHyperParameter$^{op}$ back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#hasHyperParameter

A relation between an implementation of a machine learning algorithm and its hyperparameter.

**has super-properties**
top object property

## hasInput$^{op}$ back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#hasInput

A relation between a run and data that is taken as input to the run.

**has super-properties**
top object property

## hasOutput$^{op}$ back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#hasOutput

A relation between a run and either a model or model evaluation that is produced on it's output.

**has super-properties**
top object property

## hasPart$^{op}$ back to [ToC](#) or [Object Property ToC](#)

**IRI:** http://www.w3.org/ns/mls#hasPart

A relation which represents a part-whole relationship holding between an entity and its part.

**has super-properties**
top object property

## hasQuality$^{op}$ back to [ToC](#) or [Object Property ToC](#)

**IRI:** http://www.w3.org/ns/mls#hasQuality

A relation between entities and their various characteristics.

**has super-properties**
top object property

## implements$^{op}$ back to [ToC](#) or [Object Property ToC](#)

**IRI:** http://www.w3.org/ns/mls#implements

A relation between an information entity and a specification that it conforms to.

**has super-properties**
: top object property

### realizes^op back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#realizes

A relation between a run and a task, where the run realizes specifications formulated by the task.

**has super-properties**
: top object property

### specifiedBy^op back to [ToC]() or [Object Property ToC]()

**IRI:** http://www.w3.org/ns/mls#specifiedBy

A relation between an entity and the information content entity that specifies it.

**has super-properties**
: top object property

## Annotation Properties

## description[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/description

## has version[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/hasVersion

## issued[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/issued

## modified[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/modified

### note[ap] back to ToC or Annotation Property ToC

**IRI:** http://www.w3.org/2004/02/skos/core#note

### publisher[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/publisher

### title[ap] back to ToC or Annotation Property ToC

**IRI:** http://purl.org/dc/terms/title

## 3. The relationship of ML Schema to other machine learning and data mining ontologies

In this section, we present the relationship of the MLSchema to other proposed ontologies and vocabularies for the domain of machine learning and data mining. The development of MLSchema was highly influenced from, initially independent, research of several groups on modeling the machine learning/data mining domain. Due to this the classes and relations present in the ML Schema re-appear in the current ML/DM ontologies and vocabularies. In Table 2, we present the mapping between the terms present in the MLSchema and the current ML/DM ontologies and vocabularies.

Table 2: Mapping between the terms between the ML Schema and the different

| Term from ML Schema | OntoDM-core [OntoDM-core] | DMOP [DMOP] | Expose [Expose] |
|---|---|---|---|
| Task | Data mining task | DM-Task | Task |
| Algorithm | Data mining algorithm | DM-Algorithm | Algorithm |
| Software | Data mining software | DM-Software | N/A |
| Implementation | Data mining algorithm implementation | DM-Operator | Algorithm implementatic |
| HyperParameter | Parameter | Parameter | Parameter |
| HyperParameterSetting | Parameter setting | OpParameterSetting | Parameter setting |
| Study | Investigation | N/A | N/A |
| Experiment | N/A | DM-Experiment | Experiment |
| Run | Data mining algorithm execution | DM-Operation | Algorithm execution |
| Data | Data item | DM-Data | N/A |
| Dataset | DM dataset | DataSet | Dataset |
| Feature | N/A | Feature | N/A |
| DataCharacteristic | Data | DataCharacteristic | Dataset |

| | | | |
|---|---|---|---|
| | specification | | specification |
| DatasetCharacteristic | Dataset specification | DataSetCharacteristic | Data quality |
| FeatureCharacteristic | Feature specification | FeatureCharacteristic | N/A |
| Model | Generalization | DM-Hypothesis (DM-Model / DM-PatternSet) | Model |
| ModelCharacteristic | Generalization quality | HypothesisCharacteristic | Model Structure, Parameter, ... |
| ModelEvaluation | Generalization evaluation | ModelPerformance | Evaluation |
| EvaluationMeasure | Evaluation datum | ModelEvaluationMeasure | Evaluation measure |
| EvaluationSpecification | N/A | N/A | N/A |
| EvaluationProcedure | Evaluation algorithm | ModelEvaluationAlgorithm | Performance Estimation |

# The OntoDM-core ontology

For the domain of data mining there are several developed ontologies, with the aim of providing formal descriptions of domain entities. One of the proposed ontologies is the OntoDM-core ontology **[OntoDM-core]**. In one of the preliminary versions of the ontology **[OntoDM-core-init]**, the authors decided to align the proposed ontology with the Ontology of Biomedical Investigations (OBI) **[OBI]** and consequently with the Basic Formal Ontology (BFO) at the top level **[BFO]**, in terms of top-level classes and the set of relations.

That was beneficial for structuring the domain in a more elegant way and the basic differentiation of information entities, implementation entities and processual entities. In this context, the authors proposed a horizontal description structure that includes three layers: a specification layer, an implementation layer, and an application layer. The specification layer in general contains information entities. In the domain of data mining, example classes are *data mining task* and *data mining algorithm*. The implementation layer in general contains qualities and entities that are realized in a process, such as parameters and implementations of algorithms. The application layer contains processual classes, such as the execution of the data mining algorithm.

## The Exposé ontology

The main goal of **[EXPOSE]** is to describe (and reason about) machine learning experiments. It is built on top of OntoDM, as well as top-level ontologies from bio-informatics. It is currently used in OpenML, as a way to structure data (e.g. database design) and share data (APIs). MLSchema will be used to export all OpenML data as linked open data (in RDF).

## The DMOP ontology

The Data Mining OPtimization Ontology (DMOP) **[DMOP]** has been developed with a primary use case in meta-mining, that is meta-learning extended to an analysis of full DM processes. At the level of both single algorithms and more complex workflows, it follows a very similar modeling pattern as described in the MLSchema. To support meta-mining, DMOP contains a taxonomy of algorithms used in DM processes which are described in detail in terms of their underlying assumptions, cost functions, optimization strategies, generated models or pattern sets, and other properties. Such a "glass box" approach which makes explicit internal algorithm characteristics allows meta-learners using DMOP to generalize over algorithms and their properties, including those algorithms which were not used for training meta-learners.

## The MEX vocabulary

The MEX vocabulary has been designed to reuse existing ontologies (i.e., **[PROV-O]**, **[DUBLIN-CORE]**, and **[DOAP]**) for representing basic machine learning information. The aim is not to describe a complete data-mining process, which can be modeled by more complex and semantically refined structures. Instead, MEX is designed to provide a simple and lightweight vocabulary for exchanging machine learning metadata in order to achieve a high level of interoperability as well as supporting data management for ML outcomes.

# 4. Acknowledgements

This section is non-normative.
The editors of this document thank the developers of the tools that were used to create the ML Schema and parts of this document.

Those tools were of major help for developing ML Schema as well as the documentation.

- Protégé group from the Stanford University for providing a free, open-source ontology editor and framework for building intelligent systems.
- Silvio Peroni for LODE tool for the CSS styling of parts of this page.
- Astah Community Edition developers for providing a tool for visualizing our models in UML.
- Robin Berjon for providing ReSpec 3 CSS for styling of parts of this page.

## 5. References


**[BFO]**

Arp, R., Smith, B., Spear, A.D. *Building Ontologies with Basic Formal Ontology*. The MIT Press (2015)

**[DMOP]**

C. Maria Keet, Agnieszka Ławrynowicz, Claudia d'Amato, Alexandros Kalousis, Phong Nguyen, Raul Palma, Robert Stevens, Melanie Hilario, *Data Mining OPtimization Ontology*. Web Semantics: Science, Services and Agents on the World Wide Web (2015) 32:43-53.

**[DUBLIN-CORE]**

Dublin Core Metadata Terms *http://dublincore.org/documents/dcmi-terms/*

**[DOAP]**
DOAP *https://github.com/edumbill/doap/wiki*

**[EXPOSE]**
Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G, *Experiment databases - A new way to share, organize and learn from experiments* . Machine Learning (2012) 87(2):127-158.

**[FOAF]**
Dan Brickley, Libby Miller. *FOAF Vocabulary Specification 0.98*. 9 August 2010. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

**[MEX]**
Esteves, D., Moussallem, D., Baron Neto, C., Soru, T., Usbeck, R., Ackermann, M., Lehmann, J. *MEX vocabulary: a lightweight interchange format for machine learning experiments*. ACM (2015).

**[OBI]**
Bandrowski, A., Brinkman, R., Brochhausen, M., Brush, M.H., Bug, B., Chibucos, M.C., Clancy, K., Courtot, M., Derom, D., Dumontier, M., Fan, L., Fostel, J., Fragoso, G., Gibson, F., Gonzalez-Beltran, A., Haendel, M.A., He, Y., Heiskanen, M., Hernandez-Boussard, T., Jensen, M., Lin, Y., Lister, A.L., Lord, P., Malone, J., Manduchi, E., McGee, M., Morrison, N., Overton, J.A., Parkinson, H., Peters, B., Rocca-Serra, P., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Schober, D., Smith, B., Soldatova, L.N., Stoeckert, Jr., C.J., Taylor, C.F., Torniai, C., Turner, J.A., Vita, R., Whetzel, P.L., Zheng, J. *The ontology for biomedical investigations*. PLoS ONE (2016) 11(4): e0154556.

**[OntoDM-core]**

Panov, P., Soldatova, L. & Džeroski, S. *Ontology of core data mining entities*. Data Min Knowl Disc (2014) 28: 1222-1265.

**[OntoDM-core-init]**

Panov, P., Soldatova, L. & Džeroski, S. *Towards an Ontology of Data Mining Investigations* . Lecture notes in computer science (2009) 5808: 257-271.

**[OWL2-OVERVIEW]**

W3C OWL Working Group. *OWL 2 Web Ontology Language: Overview.* 27 October 2009. W3C Recommendation. URL: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/

**[PROV-O]**

Timothy Lebo, Satya Sahoo, Deborah McGuinness, *PROV-O: The PROV Ontology* W3C Recommendation 30 April 2013, URL: https://www.w3.org/TR/prov-o/

**[RDF-CONCEPTS]**

Richard Cyganiak, David Wood, Markus Lanthaler, Editors. *RDF 1.1 Concepts and Abstract Syntax*. 25 February 2014. W3C Recommendation. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

**[XMLSCHEMA11-2]**

Henry S. Thompson; et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes.* 5 April 2012. W3C Recommendation URL: http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/