**Open Geospatial Consortium**

Submission Date: <2023-01-28>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: <[http://www.opengis.net/doc/[{doc-type/}]{standard}/{m.n}](http://www.opengis.net/def/%5b%7bdoc-type/%7d%5d%7bstandard%7d/%7bm.n%7d)>

Internal reference number of this OGC® document: 23-008

Version: 1.0.0

Editor: Peng Yue, Boyi Shangguan

OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

**Copyright notice**

Copyright © 2022 Open Geospatial Consortium  
To obtain additional rights of use, visit <https://www.ogc.org/ogc/Document>.

**License Agreement**

Permission for use/distribution of this document and any associated materials is subject to the terms of this License Agreement: <https://www.ogc.org/license>

**Warning**

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Document type:    OGC® Standard

Document subtype:    if applicable

Document stage:    Draft

Document language:  English

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

[Abstract 6](#_Toc128942720)

[Keywords 6](#_Toc128942721)

[Preface 7](#_Toc128942722)

[Security Considerations 7](#_Toc128942723)

[Submitting organizations 7](#_Toc128942724)

[Submitters 7](#_Toc128942725)

[Acknowledgments 8](#_Toc128942726)

[1. Scope 8](#_Toc128942727)

[2. Conformance 9](#_Toc128942728)

[3. Normative References 9](#_Toc128942729)

[4. Terms and definitions 9](#_Toc128942730)

[4.1 Artificial Intelligence (AI) 10](#_Toc128942731)

[4.2 Machine Learning (ML) 10](#_Toc128942732)

[4.3 Deep Learning (DL) 10](#_Toc128942733)

[4.4 Training Dataset 10](#_Toc128942734)

[4.5 Label 10](#_Toc128942735)

[4.6 Provenance 10](#_Toc128942736)

[4.7 Quality 10](#_Toc128942737)

[4.8 Earth Observation 10](#_Toc128942738)

[4.9 Scene Classification 11](#_Toc128942739)

[4.10 Object Detection 11](#_Toc128942740)

[4.11 Semantic Segmentation 11](#_Toc128942741)

[4.12 Change Detection 11](#_Toc128942742)

[4.13 3D Model Reconstruction 11](#_Toc128942743)

[5. Conventions 11](#_Toc128942744)

[5.1 Identifiers 11](#_Toc128942745)

[5.2 Abbreviated terms 11](#_Toc128942746)

[5.3 UML Notation 12](#_Toc128942747)

[6. Overview 13](#_Toc128942748)

[6.1 AI tasks for EO 13](#_Toc128942749)

[6.2 Modularization 14](#_Toc128942750)

[6.3 General modeling principles 15](#_Toc128942751)

[6.3.1 Element modeling 15](#_Toc128942752)

[6.3.2 Class Hierarchy and Inheritance of Properties and Relations 16](#_Toc128942753)

[6.3.3 Definition of the Semantics for all Classes, Properties, and Relations 16](#_Toc128942754)

[6.3.4 Data Integrity, Authenticity, and Non-repudiation 16](#_Toc128942755)

[6.4 Extending TrainingDML-AI 16](#_Toc128942756)

[7. TrainingDML-AI UML Model 17](#_Toc128942757)

[7.1 ISO dependencies 17](#_Toc128942758)

[7.2 Overview of the UML model 18](#_Toc128942759)

[7.3 AI\_TrainingDataset 20](#_Toc128942760)

[7.3.1 Requirements 21](#_Toc128942761)

[7.3.2 Class definitions 22](#_Toc128942762)

[7.4 AI\_TrainingData 23](#_Toc128942763)

[7.4.1 Requirements 24](#_Toc128942764)

[7.4.2 Class definitions 25](#_Toc128942765)

[7.5 AI\_Task 26](#_Toc128942766)

[7.5.1 Requirements 27](#_Toc128942767)

[7.5.2 Class definitions 28](#_Toc128942768)

[7.6 AI\_Label 28](#_Toc128942769)

[7.6.1 Requirements 29](#_Toc128942770)

[7.6.2 Class definitions 30](#_Toc128942771)

[7.7 AI\_Labeling 31](#_Toc128942772)

[7.7.1 Requirements 32](#_Toc128942773)

[7.7.2 Class definitions 33](#_Toc128942774)

[7.8 AI\_TDChangeset 33](#_Toc128942775)

[7.8.1 Requirements 34](#_Toc128942776)

[7.8.2 Class definitions 35](#_Toc128942777)

[7.9 AI\_DataQuality 35](#_Toc128942778)

[7.9.1 Requirements 37](#_Toc128942779)

[7.9.2 Class definitions 38](#_Toc128942780)

[8. TrainingDML-AI Data Dictionary 38](#_Toc128942781)

[8.1 ISO Classes 38](#_Toc128942782)

[8.1.1 Feature (from ISO 19107:2019) 38](#_Toc128942783)

[8.1.2 MD\_Band (from ISO 19115-1:2014) 39](#_Toc128942784)

[8.1.3 MD\_Scope (from ISO 19115-1:2014) 39](#_Toc128942785)

[8.1.4 EX\_Extent (from ISO 19115-1:2014) 39](#_Toc128942786)

[8.1.5 CI\_Citation (from ISO 19115-1:2014) 39](#_Toc128942787)

[8.1.6 DataQuality (from ISO 19157-1) 40](#_Toc128942788)

[8.1.7 QualityElement (from ISO 19157-1) 40](#_Toc128942789)

[8.2 AI\_TrainingDataset 40](#_Toc128942790)

[8.2.1 Classes 40](#_Toc128942791)

[8.3 AI\_TrainingData 43](#_Toc128942792)

[8.3.1 Classes 43](#_Toc128942793)

[8.4 AI\_Task 45](#_Toc128942794)

[8.4.1 Classes 45](#_Toc128942795)

[8.5 AI\_Label 46](#_Toc128942796)

[8.5.1 Classes 47](#_Toc128942797)

[8.6 AI\_Labeling 49](#_Toc128942798)

[8.6.1 Classes 49](#_Toc128942799)

[8.7 AI\_TDChangeset 50](#_Toc128942800)

[8.7.1 Classes 51](#_Toc128942801)

[8.8 AI\_DataQuality 52](#_Toc128942802)

[8.8.1 Classes 52](#_Toc128942803)

[Annex A Abstract Test Suite (Normative) 53](#_Toc128942804)

[A.1 Introduction 53](#_Toc128942805)

[A.2 Conformance class AI\_TrainingDataset 53](#_Toc128942806)

[A.3 Conformance class AI\_TrainingData 54](#_Toc128942807)

[A.4 Conformance class AI\_Task 55](#_Toc128942808)

[A.5 Conformance class AI\_Label 56](#_Toc128942809)

[A.6 Conformance class AI\_Labeling 57](#_Toc128942810)

[A.7 Conformance class AI\_TDChangeset 58](#_Toc128942811)

[A.8 Conformance class AI\_DataQuality 59](#_Toc128942812)

[Annex B Example (Informative) 61](#_Toc128942813)

[B.1 TrainingDataset encoding examples 61](#_Toc128942814)

[WHU-RS19 dataset 61](#_Toc128942815)

[DOTA-v1.5 dataset 61](#_Toc128942816)

[KITTI 2D object detection dataset 61](#_Toc128942817)

[GID dataset 62](#_Toc128942818)

[Toronto3D dataset 62](#_Toc128942819)

[WHU-Building dataset 62](#_Toc128942820)

[California change detection dataset 62](#_Toc128942821)

[WHU MVS dataset 63](#_Toc128942822)

[B.2 DataQuality encoding example 63](#_Toc128942823)

[WHU-RS19 data quality 63](#_Toc128942824)

[B.3 TDChangeset encoding example 63](#_Toc128942825)

[DOTA-v1.5 changeset 63](#_Toc128942826)

[Annex C Revision history (Informative) 64](#_Toc128942827)

[Annex D Bibliography 65](#_Toc128942828)

Abstract

The Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Standard aims to develop the UML model and encodings for geospatial machine learning training data. Training data plays a fundamental role in Earth Observation (EO) Artificial Intelligence Machine Learning (AI/ML), especially Deep Learning (DL). It is used to train, validate, and test AI/ML models. This Standard defines a UML model and encodings consistent with the OGC Standards baseline to exchange and retrieve the training data in the Web environment.

The TrainingDML-AI Standard provides detailed metadata for formalizing the information model of training data. This includes but is not limited to the following aspects:

* How the training data is prepared, such as provenance or quality;
* How to specify different metadata used for different ML tasks such as scene/object/pixel levels;
* How to differentiate the high-level training data information model and extended information models specific to various ML applications;
* How to introduce external classification schemes and flexible means for representing ground truth labeling.

Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, artificial intelligence, machine learning, deep learning, earth observation, remote sensing, training data, training sample, UML

Preface

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Security Considerations

No security considerations have been made for this Standard.

Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

* Wuhan University
* Pixalytics Ltd
* National Geospatial-Intelligence Agency
* George Mason University
* Laboratoire d'Informatique de Grenoble
* South Digital Technology Co., Ltd
* Wuhan University of Technology
* Hubei University
* Chongqing Changan Automobile Co., Ltd

Submitters

All questions regarding this submission should be directed to the editor or the submitters:

|  |  |
| --- | --- |
| Name | Affiliation |
| Peng Yue | Wuhan University |
| Jianya Gong | Wuhan University |
| Ruixiang Liu | Wuhan University |
| Dayu Yu | Wuhan University |
| Samantha Lavender | Pixalytics Ltd |
| Jim Antonisse | National Geospatial-Intelligence Agency |
| Liping Di | George Mason University |
| Eugene Yu | George Mason University |
| Danielle Ziébelin | Laboratoire d’Informatique de Grenoble |
| Boyi Shangguan | South Digital Technology Co., Ltd |
| Lei Hu | South Digital Technology Co., Ltd |
| Liangcun Jiang | Wuhan University of Technology |
| Mingda Zhang | Hubei University |
| Kai Yan | Chongqing Changan Automobile Co., Ltd |

Acknowledgments

Thanks to the members of the TrainingDML-AI Standards Working Group of the OGC as well as all contributors of change requests and comments. In particular: Scott Simmons, Carl Reed, Ivana Ivánová, Emily Daemen, Jon Duckworth, Zheheng Liang, Jibo Xie, Yuqi Bai, Winnie Shiu, Ignacio Correas, Chenxiao Zhang, Zhipeng Cao, Haofeng Tan, Yinyin Pan, Hanwen Xu, Shuaiqi Liu, Hao Li, Ming Wang, Kaixuan Wang, Haipeng Deng.

# Scope

Training data is the building block of machine learning models. These models now constitute the majority of machine learning applications in Earth science. Training data is used to train AI/ML models, and to then validate model results. Formalizing and documenting the training data by characterizing the training data content, metadata, data quality, and provenance, and so forth is essential.

This OGC Training Data Standard (draft) describes work actions around training data:

* Documents the UML model with a target of maximizing the interoperability and usability of geospatial training data;
* Defines different AI/ML tasks and labels in earth observation, including scene level, object level and pixel level tasks;
* Describes the description of the permanent identifier, version, license, training data size, measurement or imagery used for annotation, and so on;
* Defines the description of quality (e.g., training data errors, training data unrepresentativeness) and the provenance (labeling, labeler, and labeling procedure).

# Conformance

This TrainingDML-AI Standard defines a conceptual model that is independent of any encoding or formatting technologies. The standardization targets for this Standard is:

* TrainingDML-AI Conceptual Model

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site[[1]](#footnote-1).

All requirements-classes and conformance-classes described in this document are owned by the standard identified.

# Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

[ISO 19107:2019 Geographic information — Spatial schema](https://www.iso.org/standard/66175.html)

[ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals](https://www.iso.org/standard/53798.html)

[ISO 19157-1 Geographic information — Data quality — Part 1: General requirements](https://www.iso.org/standard/78900.html)

# Terms and definitions

This document used the terms defined in [OGC Policy Directive 49](https://portal.ogc.org/public_ogc/directives/directives.php), which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

For the purposes of this document, the following additional terms and definitions apply.

## Artificial Intelligence (AI)

refers to a set of methods and technologies that can empower machines or software to learn and perform tasks like humans.

## Machine Learning (ML)

is an important branch of artificial intelligence. ML processes create models from training data by using a set of learning algorithms, and then can use these models to make predictions. Depending on whether the training data include labels, the learning algorithms can be divided into supervised and unsupervised learning.

## Deep Learning (DL)

is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

SOURCE: <https://machinelearningmastery.com/what-is-deep-learning>

## Training Dataset

a collection of samples, often labelled in terms of supervised learning. A training dataset can be divided into training, validation, and test sets. Training samples are different from samples in [OGC Observations & Measurements (O&M)](https://portal.ogc.org/files/?artifact_id=41579). They are often collected in purposive ways that deviate from purely probability sampling, with known or expected results labelled as values of a dependent variable for generating a trained predictive model.

## Label

refers to known or expected results annotated as values of a dependent variable in training samples. A training sample label is different from those on a geographical map, which are known as map labels or annotations.

## Provenance

information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness. In this standard provenance is a record of how training data were prepared.

SOURCE: W3C (https://www.w3.org/TR/prov-overview/)

## Quality

degree to which a set of inherent characteristics fulfils requirements [ISO 9000:2005, definition3.1.1]. Quality of training data (such as data imbalance and mislabeling) can impact the performance of AI/ML models.

## Earth Observation

data and information collected about our planet, whether atmospheric, oceanic or terrestrial. This includes space-based or remotely-sensed data, as well as ground-based or in situ data.

SOURCE: GEO (<https://earthobservations.org/geo_wwd.php>)

## Scene Classification

task of identifying scene categories of images, on the basis of a training set of images whose scene categories are known.

## Object Detection

task of recognizing objects such as cars from images. The objects are often localized using bounding boxes.

## Semantic Segmentation

task of assigning class labels to pixels of images or points of point clouds.

## Change Detection

task that finds the changes in an area between images taken at different times.

## 3D Model Reconstruction

task that builds 3D objects and scenes from multi-view images.

# Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

## Identifiers

The normative provisions in this specification are denoted by the URI

http://www.opengis.net/spec/TrainingDML-AI-1/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

## Abbreviated terms

In this document, the following abbreviations and acronyms are used or introduced:

AI Artificial Intelligence

DL Deep Learning

EO Earth Observation

ISO International Organization for Standardization

JSON JavaScript Object Notation

LC Land Cover

LU Land Use

ML Machine Learning

OGC Open Geospatial Consortium

RS Remote Sensing

TD Training Data

UML Unified Modelling Language

XML Extensible Markup Language

## UML Notation

The Standard is presented in this document through diagrams using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this Standard are described in the diagram in Figure 1.

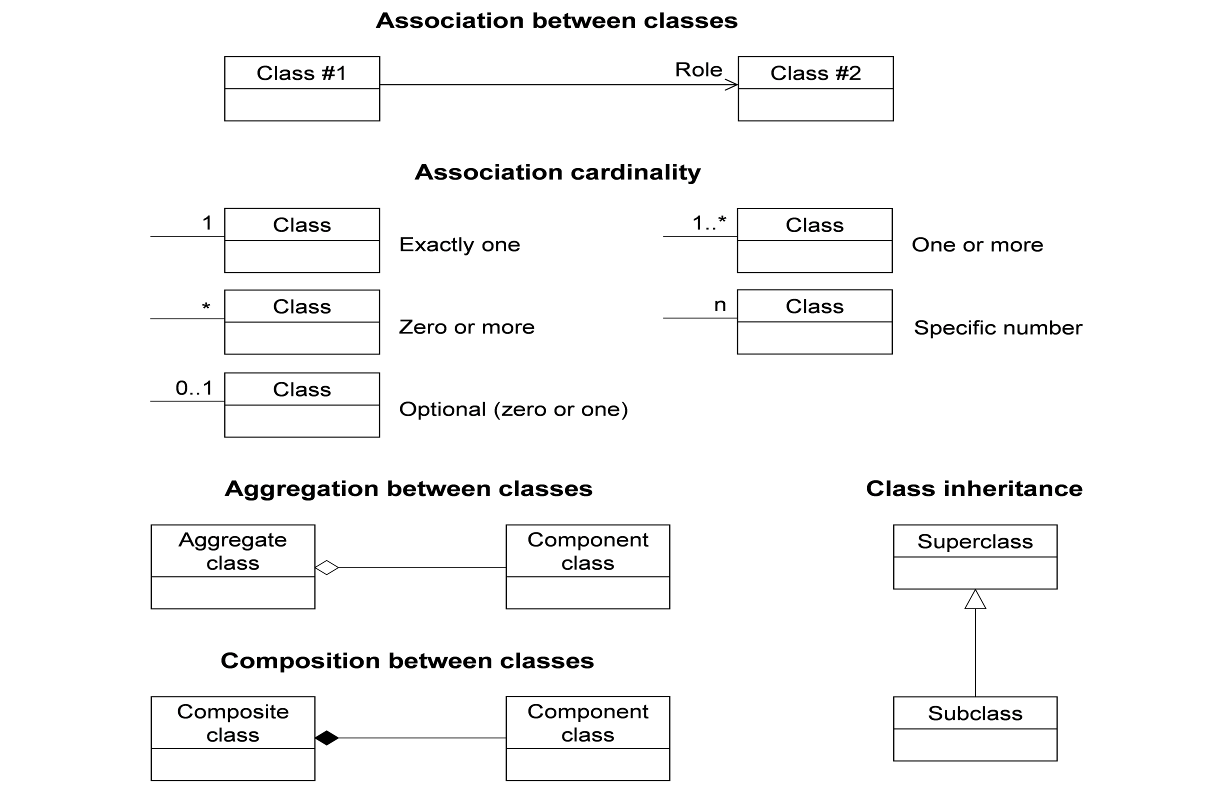


Figure 1. UML notation (see ISO TS 19103, Geographic information — Conceptual schema language).

All associations between model elements in the TrainingDML-AI Conceptual Model are uni-directional. Thus, associations in the model are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.

The following stereotypes are used in this model.

* «DataType» defines a set of properties that lack identity. A data type is a classifier with no operations, whose primary purpose is to hold information.
* «CodeList» enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline in the TrainingDML-AI UML Model. The allowed values can be provided within an external code list.

# Overview

The TrainingDML-AI Conceptual Model Standard defines how to represent and exchange ML training data. The conceptual model includes the most relevant training data entities from datasets, to instances (i.e. individual training samples), to labels. The conceptual schema specifies how and into which parts of the training data should be decomposed and classified.

The TrainingDML-AI conceptual model (Clause 7) is formally specified using UML class diagrams, complemented by a data dictionary (Clause 8) providing the definitions and explanations of the object classes and attributes. This conceptual model provides the basis for specifying encoding implemented in languages such as JSON, or XML.

## AI tasks for EO

In recent years AI/ML is increasingly used in the EO domain. The new AI/ML algorithms frequently require large training datasets as benchmarks. AI/ML TD have been used in many EO applications to calibrate the performance of AI/ML models. Many efforts have been made to produce training datasets to make accurate predictions. As a result, a number of training datasets are publicly available, with new datasets being constantly released. In the EO domain, examples of AI/ML training datasets have been developed in various tasks including the following typical scenarios:

* Scene classification. These algorithms determine image categories from numerous pictures (e.g., agricultural, forest, and beach scenes). The training samples are a series of labelled pictures. The data can be either from satellite, drones, or aircrafts. The metadata of the datasets often includes the number of training samples, the number of classes, and the image size.
* Object detection. These algorithms detect and localize different objects (e.g., airplanes, cars and building) in a single image. The image can be optical or non-optical, such as Synthetic Aperture Radar (SAR). Recent work also suggests an increasing focus on object detection from street view imagery. Objects can be labelled with two forms of bounding boxes, i.e., oriented and horizontal bounding boxes. The geometry of a bounding box can be expressed using top-left/bottom-right coordinates, coordinates of four corners, or center coordinates along with the length and width of the box.
* Semantic segmentation. In terms of Land cover (LC) and land use (LU) classification, this process assigns a LC/LU class label to a pixel (or groups of pixels) of RS imagery. Considering semantic segmentation of 3D point clouds, it is to classify points of a 3D point cloud into categories. TDs are usually composed of RS images/point clouds, and the corresponding labelled value of each pixel/point recording its class.
* Change detection. These algorithms identify the difference between images acquired over the same geographical area but taken at different times. The TD comprise a set of pre-change and post-change RS images, with the corresponding ground truth map labelled changed and unchanged pixels. The image can be optical or SAR images.
* 3D model reconstruction. These algorithms infer the 3D geometry and structure of objects and scenes, mainly realized from the dense matching of multi-view images. The TD are usually composed of two-view or multi-view images, with the corresponding disparity map or depth maps as ground truth respectively.

## Modularization

The TrainingDML-AI conceptual model provides models for the most important elements within TD. These elements have been identified to be either required or important in many different AI/ML tasks. However, implementations are not required to support the complete TrainingDML-AI model in order to be conformant to the Standard. Implementations may employ a subset of constructs according to their specific information needs. For this purpose, modularization is applied to the TrainingDML-AI.

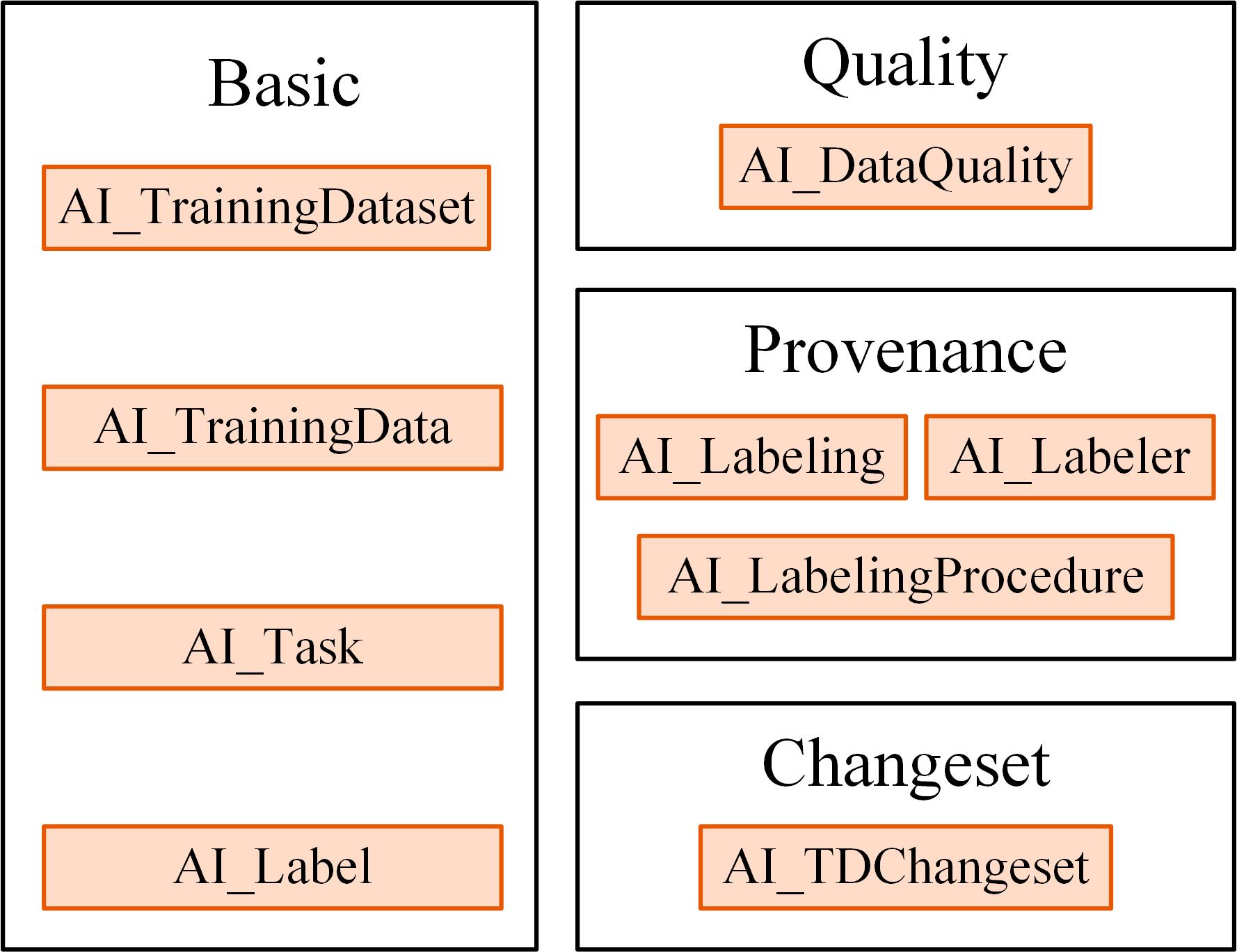


Figure 2. TrainingDML-AI module overview.

As shown in Figure 2, the TrainingDML-AI conceptual model is thematically decomposed into a Basic module, a Provenance module and a Quality module. The Basic module comprises the basic concepts and elements, including AI\_TrainingDataset, AI\_TrainingData, AI\_Label, and AI\_Task, of the TrainingDML-AI, and thus, must be implemented by any conformant system. The Provenance module provides a comprehensive definition of provenance by AI\_Labeling, AI\_Labeler, and AI\_Labeling Procedure. The Quality module offers quality description of TD with AI\_DataQuality elements. And the Changeset module defines AI\_TDChangeset between versions of datasets.

## General modeling principles

### Element modeling

The modeling of all elements in the TrainingDML-AI conceptual model has the following principles (Reference [7]):

* Granularity. Two levels of granularity are differentiated in the conceptual model: The Training Dataset is used to refer to the collection level, and the Training Data is used to refer to the individual level.
* Label semantics. The training dataset will not be limited to one classification scheme. External classification schemes should be allowed to be linked into the Training Dataset to accommodate different cases in practice.
* Light-weight design. The lightweight designed conceptual model has a minimum set of metadata elements, provenance, or quality measures at the collection level instead of at the individual level. This is to facilitate the understanding of the dataset and improve the scalability for communicating large training datasets.
* Alignment. The modelling of elements in TDs can leverage existing efforts for wide adoption, such as for ISO 19109 Geographic information — Rules for application schema, ISO 19115-1 Geographic information — Metadata — Part 1: Fundamentals, ISO 19157-1 Geographic information — Data quality — Part 1: General requirements, and the OGC Geography Markup Language (GML) Standard. The conceptual model can be aligned with these existing standards and leverage capabilities fulfilled in part by other standards.
* Quality, bias, and ethics. Elements related to quality, or more specifically, bias that can be used to reduce the errors when using AI/ML. For example, any knowledge of the TD imbalance and mislabeling can be stored in TD quality. In addition, data ethics aims to safeguard the responsible use of TD, and it can be addressed by using the license property in the TD.
* Changeset. This will be an optional module in TD modelling. Changeset addresses how to capture changes in TD datasets. The change model considers the trend in TD collections to use the crowdsourcing platforms and borrow the change representation from the platforms such as OpenStreetMap.

### Class Hierarchy and Inheritance of Properties and Relations

In the TrainingDML-AI conceptual model, the specific elements such as EO training datasets, EO training data, scene label, object label, and pixel label are defined as subclasses of more general higher-level classes. Hence, elements build a hierarchy along specialization / generalization relationships where more specialized elements inherit the properties and relationships of all their super classes along the entire generalization path to the topmost element.

### Definition of the Semantics for all Classes, Properties, and Relations

The meanings of all elements defined in the TrainingDML-AI conceptual model are normatively specified in the data dictionary in Clause 8.

### Data Integrity, Authenticity, and Non-repudiation

Sometimes training datasets can be downloaded, disseminated, and changed by anyone. The data integrity, authenticity, and non-repudiation are important to ensure unexpected bias propagation and distorted results. Currently the standard focuses on the information modelling, while data dissemination can be enriched with strategies from the general information domain by publishing hashes (e.g., MD5) and public-keys (e.g., RSA) after signing and encrypting.

## Extending TrainingDML-AI

The TrainingDML-AI conceptual model is designed as a universal information model that defines elements and attributes which are useful for a broad range of AI/ML applications. In practical AI/ML applications, the elements within specific TDs will most likely contain attributes which are not explicitly modeled in TrainingDML-AI. Moreover, there might be TD elements which are not covered by the TrainingDML-AI thematic classes.

The model provides an abstract class-based method to support the exchange of such data. Elements not represented by the predefined thematic classes of the model may be modeled and exchanged by extending abstract class.

# TrainingDML-AI UML Model

The TrainingDML-AI UML model is the normative definition of the TrainingDML-AI Conceptual Model. The tables and figures in this section were software generated from the UML model. As such, this section provides a normative representation of the TrainingDML-AI Conceptual Model.

## ISO dependencies

TrainingDML-AI builds on the ISO 19100 family of standards. The applicable standards are identified in Figure 3. Data dictionaries are included for all the ISO-defined classes explicitly referenced in the TrainingDML-AI UML model. These data dictionaries are provided for the convenience of the user. The ISO standards are the normative source.

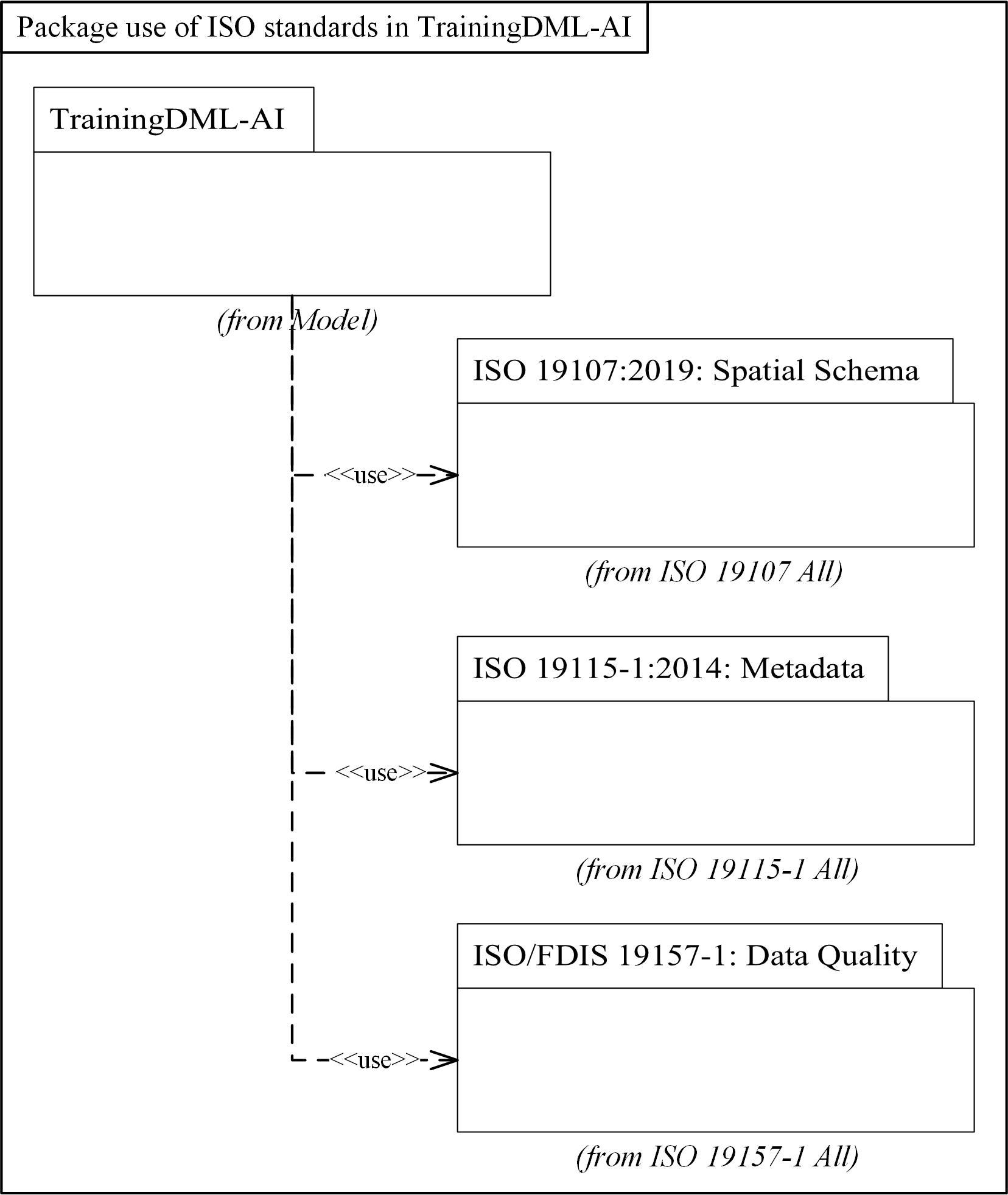


Figure 3. Use of ISO Standards in TrainingDML-AI

The ISO classes explicitly used in the TrainingDML-AI UML model are introduced in Table 1. Further details about these classes can be found in the Data Dictionary within Clause 8.

Table 1. ISO Classes used in TrainingDML-AI

|  |  |
| --- | --- |
| Class Name | Description |
| Feature | Abstraction of real-world phenomena |
| MD\_Band | Range of wavelengths in the electromagnetic spectrum. |
| MD\_Scope | The target resource and physical extent for which information is reported. |
| EX\_Extent | Extent of the resource. |
| CI\_Citation | Standardized resource reference. |
| DataQuality | Quality information for the data specified by a data quality scope. |
| QualityElement | Aspect of quantitative quality information. |

## Overview of the UML model

The UML model is presented with core concepts in Figure 4, followed by the concrete classes in Figure 5. The following describes the core concepts:

* AI\_TrainingDataset: This concept represents a collection of training samples, i.e. a training dataset.
* AI\_TrainingData: This concept is an individual training sample in a training dataset.
* AI\_Task: This concept is used to identify the task that the training dataset is used for.
* AI\_Label: This concept represents the label semantics for TD.
* AI\_Labeling: This concept provides the provenance of how TD are created.
* AI\_TDChangeset: This concept records types of TD changes between two versions of the training dataset.
* AI\_DataQuality: This concept is associated with a training dataset to document its quality.

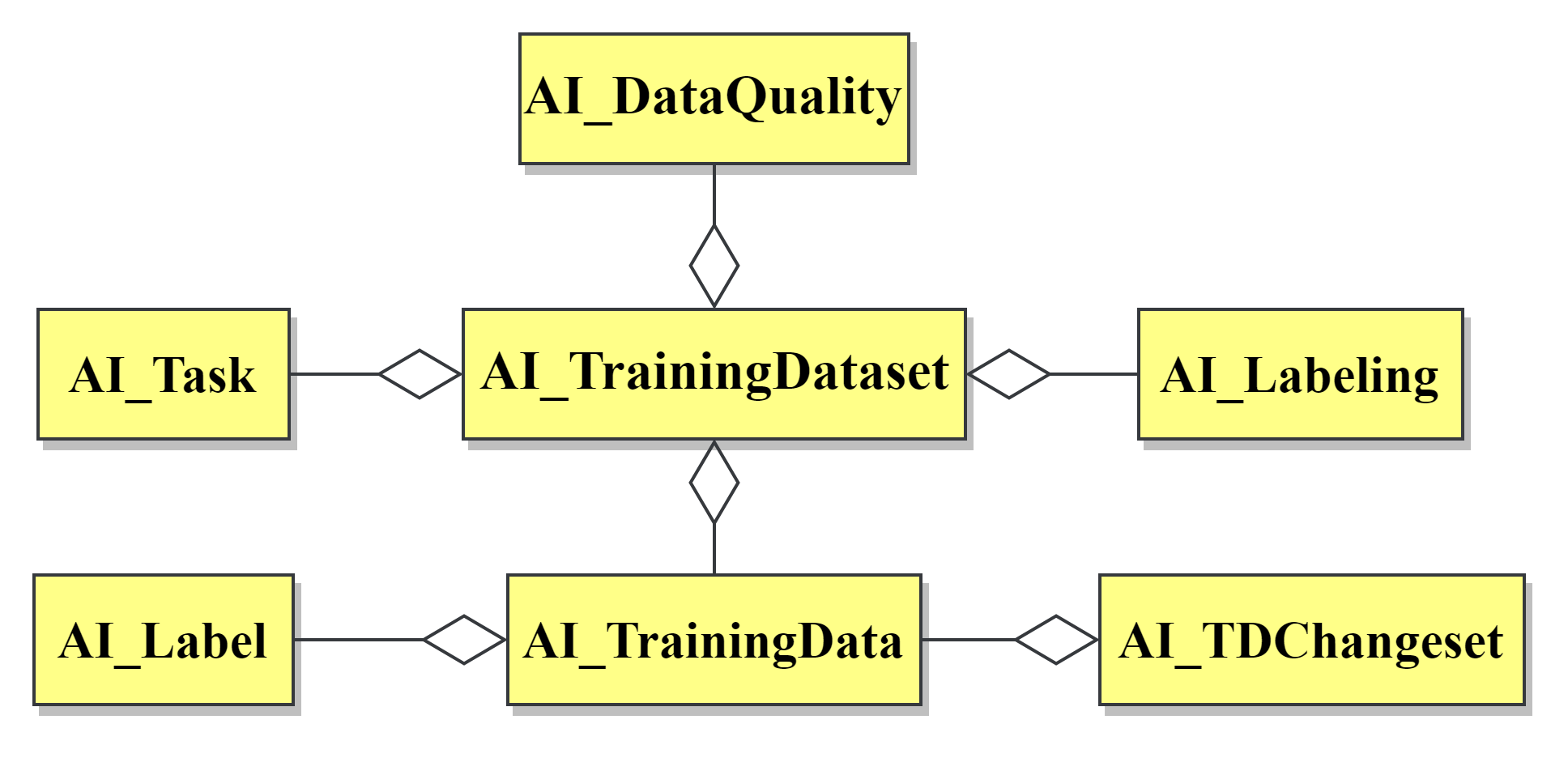
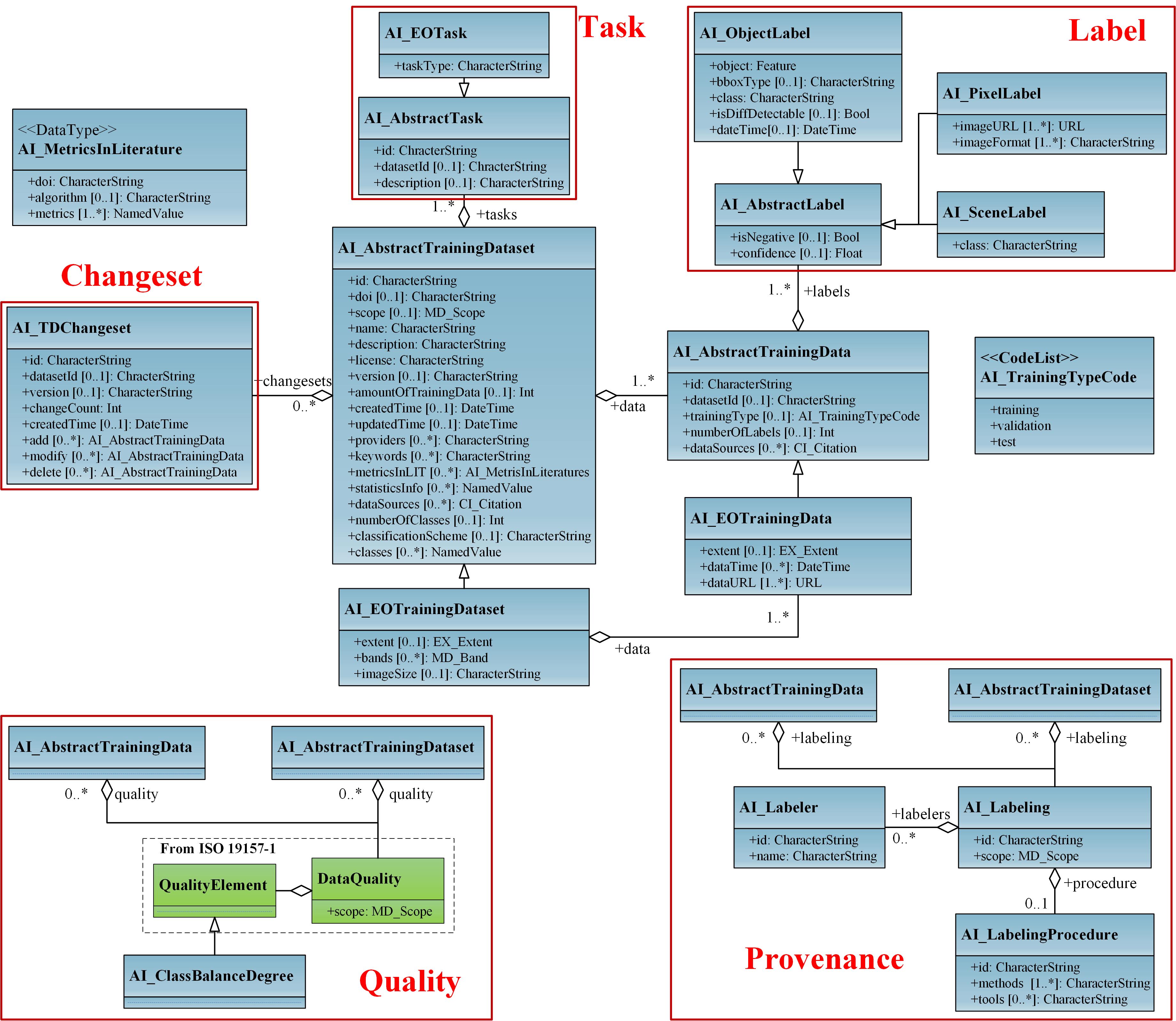


Figure 4. Core concepts.

The full overview of concrete classes and attributes are presented in Figure 5. Concepts related to the EO AI/ML applications are defined as classes extended from abstract classes. Each core concept with related classes will be described in the rest subsections.

Figure 5. Overview of the UML model.

## AI\_TrainingDataset

|  |  |
| --- | --- |
| **Requirements class 1** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-aitrainingdataset | |
| Target type | Implementation Standard |
| Dependency | ISO 19115-1:2014 |
| Dependency | /req/req-class-aitrainingdata |
| Dependency | /req/req-class-aitask |
| Dependency | /req/req-class-ailabeling |
| Dependency | /req/req-class-aidataquality |
| Dependency | /req/req-class-aitdchangeset |

An AI training dataset is represented as AI\_AbstractTrainingDataset. Each training sample in the dataset is represented as AI\_AbstractTrainingData. A set of basic attributes are defined at the collection level for AI\_AbstractTrainingDataset. These include the identification, authorship, time for creation and update, AI tasks, number of training samples (amountOfTrainingData), label semantics (classes recording the available semantic classes), number of label classes, and statistics of training samples in each class (statisticsInfo). These basic attributes at the collection level can also be seen as a minimum set of metadata to define a simple payload. In this context, payload means the actual content that will be transmitted on the Web. They can be mapped to the MD\_Metadata entity properties in ISO 19115, and thus delegate the metadata description via a metadata association to existing metadata standards.

The AI\_EOTrainingDataset is defined to convey attributes specific to EO. For example, the data source (e.g., EO images), image size, band information, and spatial extent are defined in AI\_EOTrainingDataset.

The UML diagram of the AI\_TrainingDataset is illustrated in Figure 6.

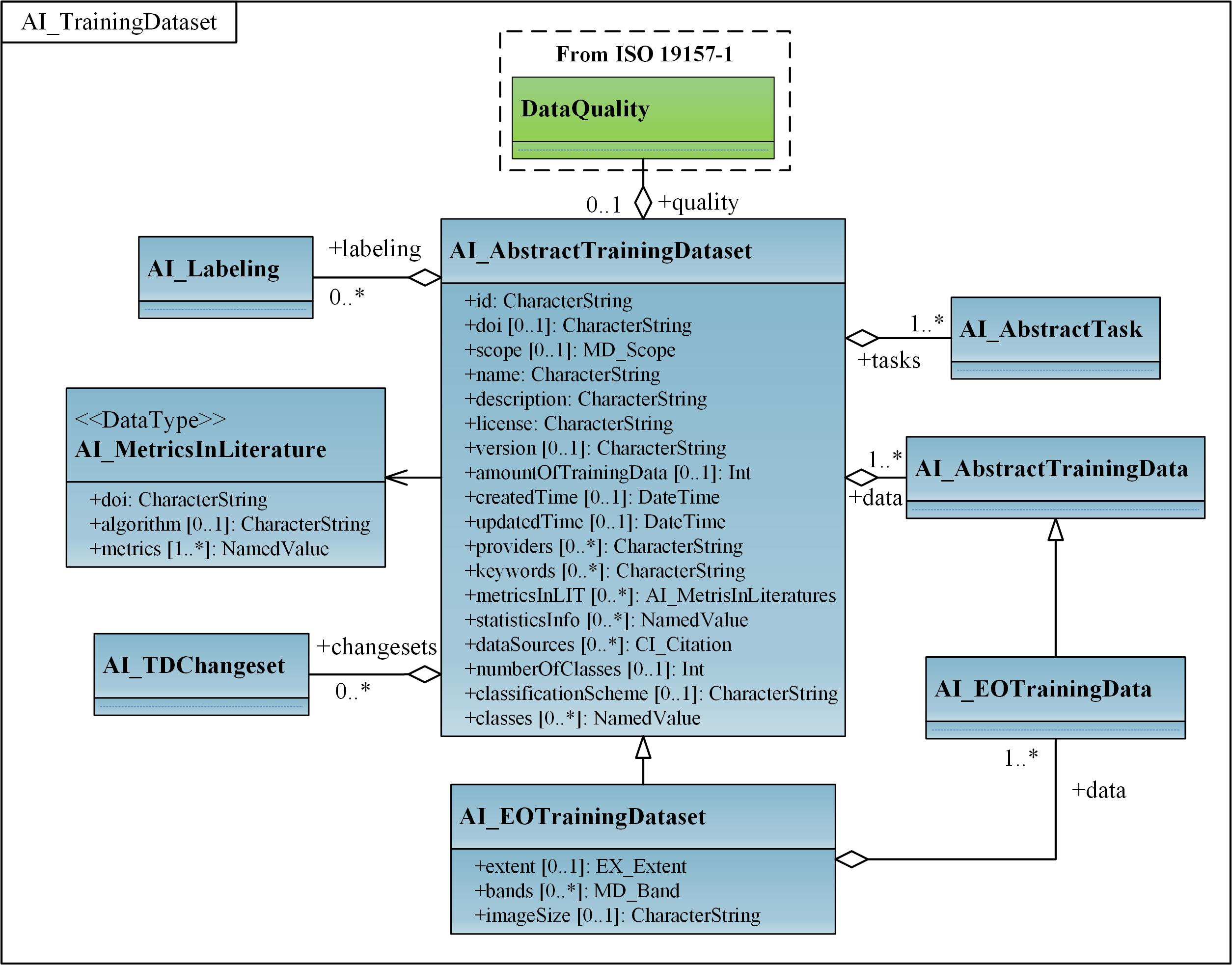


Figure 6. UML diagram of AI\_TrainingDataset

### Requirements

The following requirement defines the rules governing the implementation of the AI\_TrainingDataset conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 1** | |
| /req/aitrainingdataset/classes | |
| For each UML class defined or referenced in AI\_TrainingDataset conceptual model | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

An implementing technology may not be able to support all the concepts defined in the TrainingDML-AI UML model. Alternately, some concepts may be inappropriate for the application domain for which the implementation standard was developed. In those cases, elements of the UML model may be mapped to null elements.

|  |  |
| --- | --- |
| **Permission 1** | |
| /per/aitrainingdataset/classes | |
| For each UML class defined or referenced in AI\_TrainingDataset | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_TrainingDataset, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 2. Classes defined in AI\_TrainingDataset

|  |  |
| --- | --- |
| Name | Description |
| AI\_AbstractTrainingDataset | AI\_AbstractTrainingDataset defines the basic concepts and components of different kinds of training dataset. |
| AI\_EOTrainingDataset | AI\_EOTrainingDataset describes attributes specific to EO training dataset. For example, the data source, image size, band information, and spatial extent. |

Table 3. Data types defined in AI\_TrainingDataset

|  |  |
| --- | --- |
| Name | Description |
| AI\_MetricsInLiterature  <<DataType>> | AI\_MetricsInLiterature records results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature. |

## AI\_TrainingData

|  |  |
| --- | --- |
| **Requirements class 2** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-aitrainingdata | |
| Target type | Implementation Standard |
| Dependency | ISO 19115-1:2014 |
| Dependency | /req/req-class-aitrainingdataset |
| Dependency | /req/req-class-ailabel |

The AI\_AbstractTrainingData includes the source data and its corresponding labels, which focus on a compact modelling of individual training samples. When there is a need to identify the different uses/purposes, an optional attribute, the training type (training, validation, or test types), can be added at the individual level. The attribute datasetId helps identify samples from different training datasets.

The AI\_EOTrainingData is defined to convey attributes specific to EO. It provides additional attributes, including spatial extent and date/time of the source EO inputs.

The UML diagram of the AI\_TrainingData is illustrated in Figure 7.

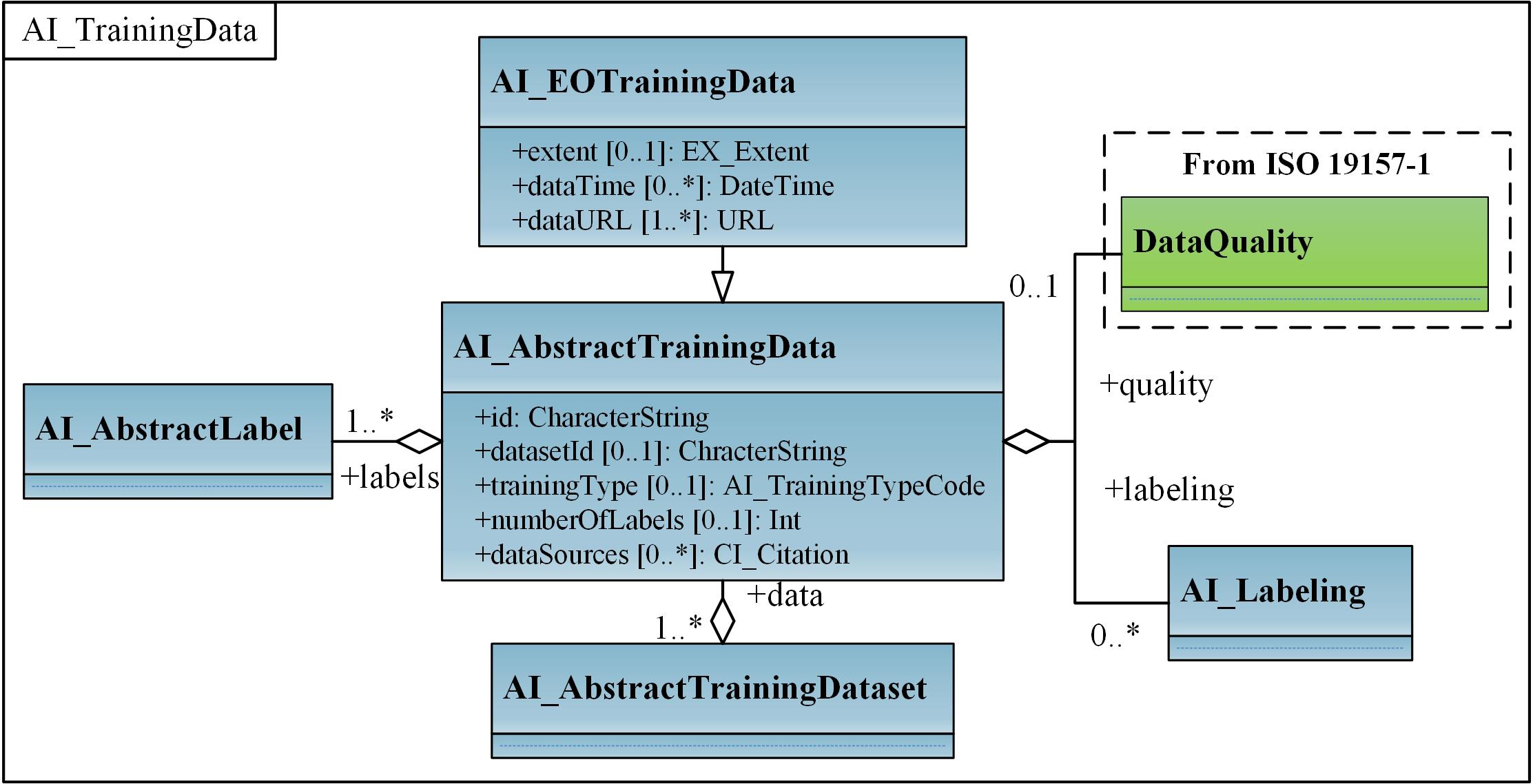


Figure 7. UML diagram of AI\_TrainingData

The Code Lists provided for the AI\_TrainingData are illustrated in Figure 8.

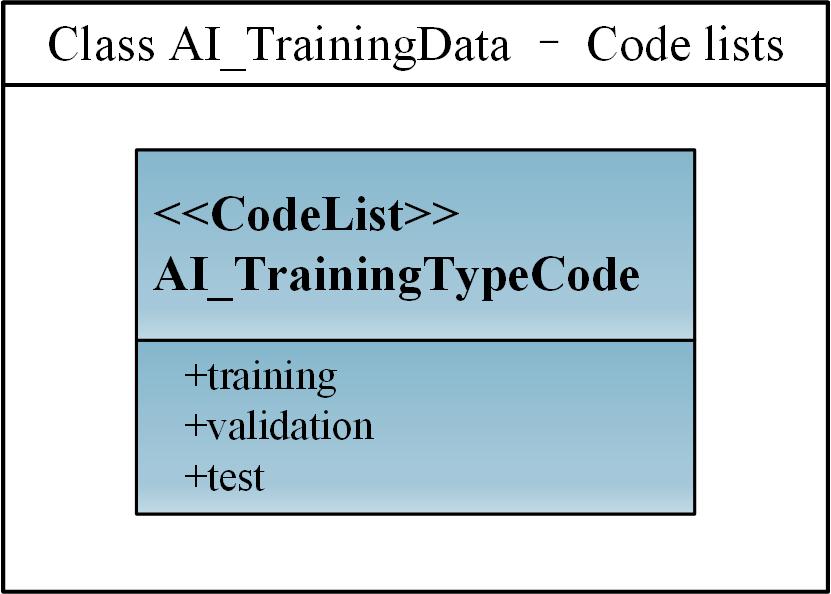


Figure 8. Code lists from the AI\_TrainingData

### Requirements

The following requirement defines the rules governing the implementation of the AI\_TrainingData conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 2** | |
| /req/aitrainingdata/classes | |
| For each UML class defined or referenced in AI\_TrainingData | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class, including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class, including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class, including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 2** | |
| /per/aitrainingdata/classes | |
| For each UML class defined or referenced in AI\_TrainingData | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_TrainingData, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 4. Classes defined in AI\_TrainingData

|  |  |
| --- | --- |
| Name | Description |
| AI\_AbstractTrainingData | AI\_AbstractTrainingData defines the basic concepts and components of different kinds of training data. |
| AI\_EOTrainingData | AI\_EOTrainingData describes attributes specific to EO training data. For example, spatial extent and date time of the source EO inputs. |

Table 5. Code list classes defined in AI\_TrainingData

|  |  |
| --- | --- |
| Name | Description |
| AI\_TrainingTypeCode  <<CodeList>> | AI\_TrainingTypeCode is a code list used to identify whether the individual data element is used for different purposes. |

## AI\_Task

|  |  |
| --- | --- |
| **Requirements class 3** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-aitask | |
| Target type | Implementation Standard |
| Dependency | /req/req-class-aitrainingdataset |

Various AI/ML tasks in specific domains can have different organizing forms of TD, including source data and label representations. AI\_EOTask is proposed by extending AI\_AbstractTask to represent specific AI/ML tasks in the EO domain. The task type can refer to a particular type defined by an external category, such as scene classification or change detection. It specifies, for example, whether a remote sensing training dataset is used for scene classification, objection detection, land use or land cover, or change detection scenarios.

The UML diagram of the AI\_Task is illustrated in Figure 9.

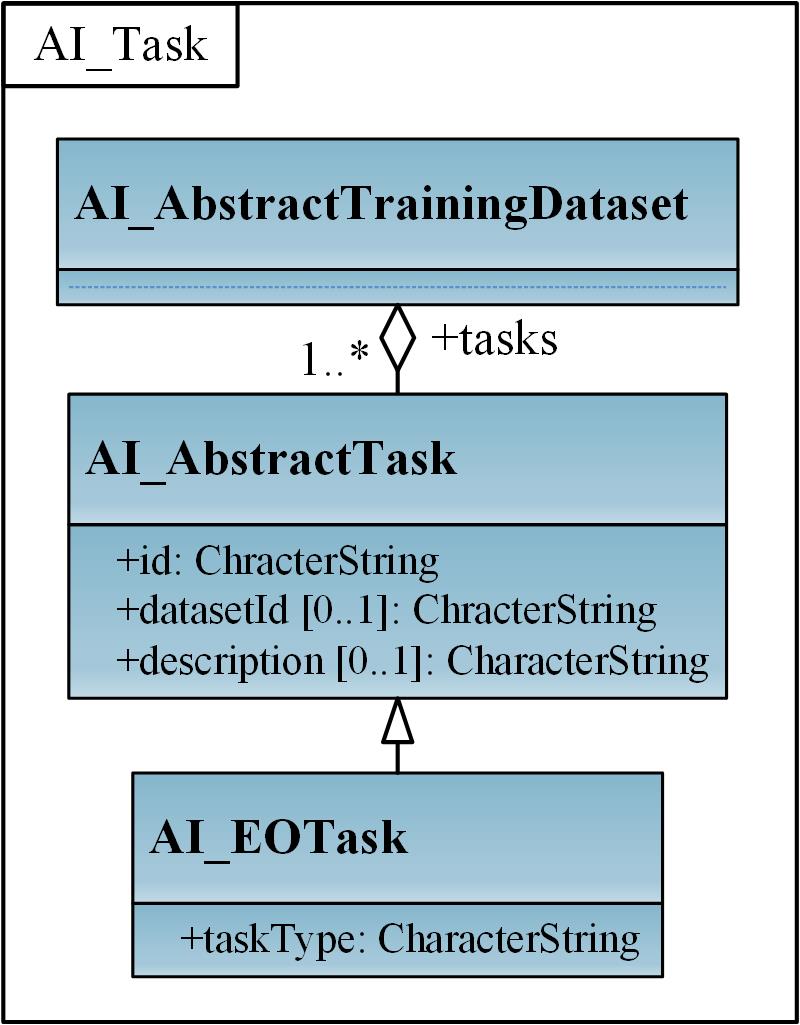


Figure 9. UML diagram of AI\_Task

### Requirements

The following requirement defines the rules governing the implementation of the AI\_Task conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 3** | |
| /req/aitask/classes | |
| For each UML class defined or referenced in AI\_Task | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 3** | |
| /per/aitask/classes | |
| For each UML class defined or referenced in AI\_Task | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_Task, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 6. Classes defined in AI\_Task

|  |  |
| --- | --- |
| Name | Description |
| AI\_AbstractTask | AI\_AbstractTask defines the AI/ML tasks in specific domains that have different organizing forms of TD. |
| AI\_EOTask | AI\_EOTask extends AI\_AbstractTask to represent specific AI/ML tasks in the EO domain. |

## AI\_Label

|  |  |
| --- | --- |
| **Requirements class 4** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-ailabel | |
| Target type | Implementation Standard |
| Dependency | /req/req-class-aitrainingdata |

Labels for each individual training sample can be represented using a feature, coverage, or a semantic class from ontologies or shared vocabularies (external classification schemes and classes). A training sample typically relates pixels, objects, and scenes to semantic labels, where labels are encoded as coverage, geometric polygon, and text respectively. Therefore, the AI\_AbstractLabel is extended to specify AI\_SceneLabel, AI\_ObjectLabel, and AI\_PixelLabel respectively. There is no restriction for TD producers to describe label information with label classes. For example, labels of training dataset for object detection can be represented by either AI\_ObjectLabel or AI\_PixelLabel.

The UML diagram of the AI\_Label is illustrated in Figure 10.

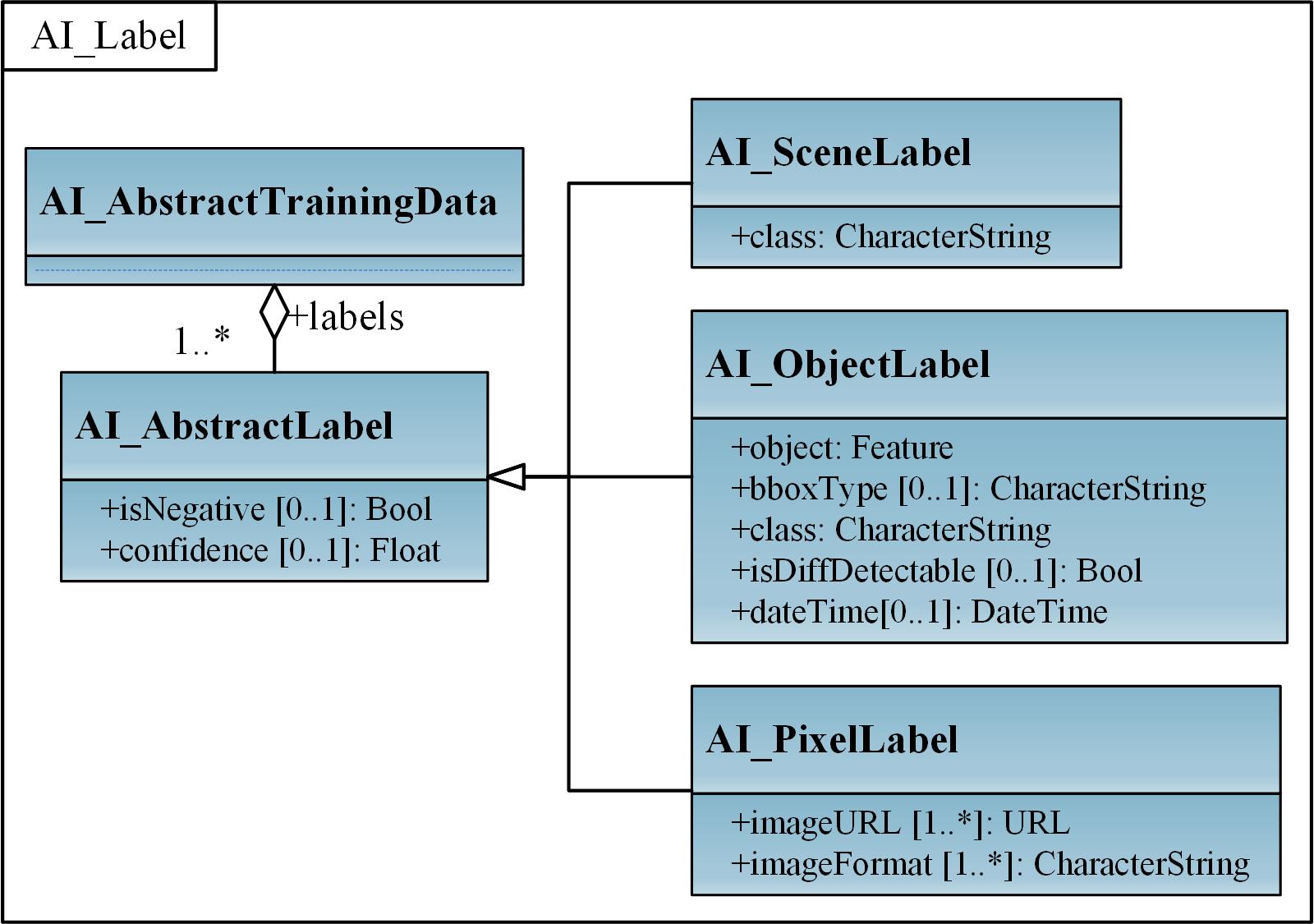


Figure 10. UML diagram of AI\_Label

### Requirements

The following requirement defines the rules governing the implementation of the AI\_Label conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 4** | |
| /req/ailabel/classes | |
| For each UML class defined or referenced in AI\_Label | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 4** | |
| /per/ailabel/classes | |
| For each UML class defined or referenced in AI\_Label | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_Label, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 7. Classes defined in AI\_Label

|  |  |
| --- | --- |
| Name | Description |
| AI\_AbstractLabel | AI\_AbstractLabel defines a set of informative tags or metadata obtained by labelling training data. It can be represented using a feature, coverage, or a semantic class from ontologies or shared vocabularies. |
| AI\_SceneLabel | AI\_SceneLabel represents the scene level label using a semantic from ontologies or shared vocabularies. |
| AI\_ObjectLabel | AI\_ObjectLabel represents the object level label using a feature. |
| AI\_PixelLabel | AI\_PixelLabel represents the pixel level label using a coverage. |

## AI\_Labeling

|  |  |
| --- | --- |
| **Requirements class 5** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-ailabeling | |
| Target type | Implementation Standard |
| Dependency | /req/req-class-aitrainingdataset |

AI\_Labeling can be associated with AI\_AbstractTrainingDataset or AI\_AbstractTrainingData to record basic provenance information on how the training dataset or training samples are created. For example, AI\_LabelingProcedure can be used to record the sampling strategy, and dataSources from AI\_AbstractTrainingDataset/ AI\_AbstractTrainingData can refer to any additional geospatial data that was used as part of the sampling strategy.

AI\_Labeling includes the labeler and the labeling procedure, which can be mapped to the agent and activity respectively in [W3C PROV](https://www.w3.org/TR/prov-overview/) model. The labeler identifies the agent that creates the training dataset or individual samples, and the labeling procedure represents the process for data generation. AI\_Labeling can also describe how the input data of the TD has been manipulated, such as resampled, color corrected, atmospherically corrected, and terrain corrected. For example, imagery with 10cm and 15cm resolution was resampled to 20cm prior for labeling.

Alternatively, despite the simple payload in this TrainingDML-AI Standard, providing a comprehensive definition of provenance using the ISO 19115 lineage model through a metadata association in the data is also applicable.

The UML diagram of the AI\_Labeling is illustrated in Figure 11.

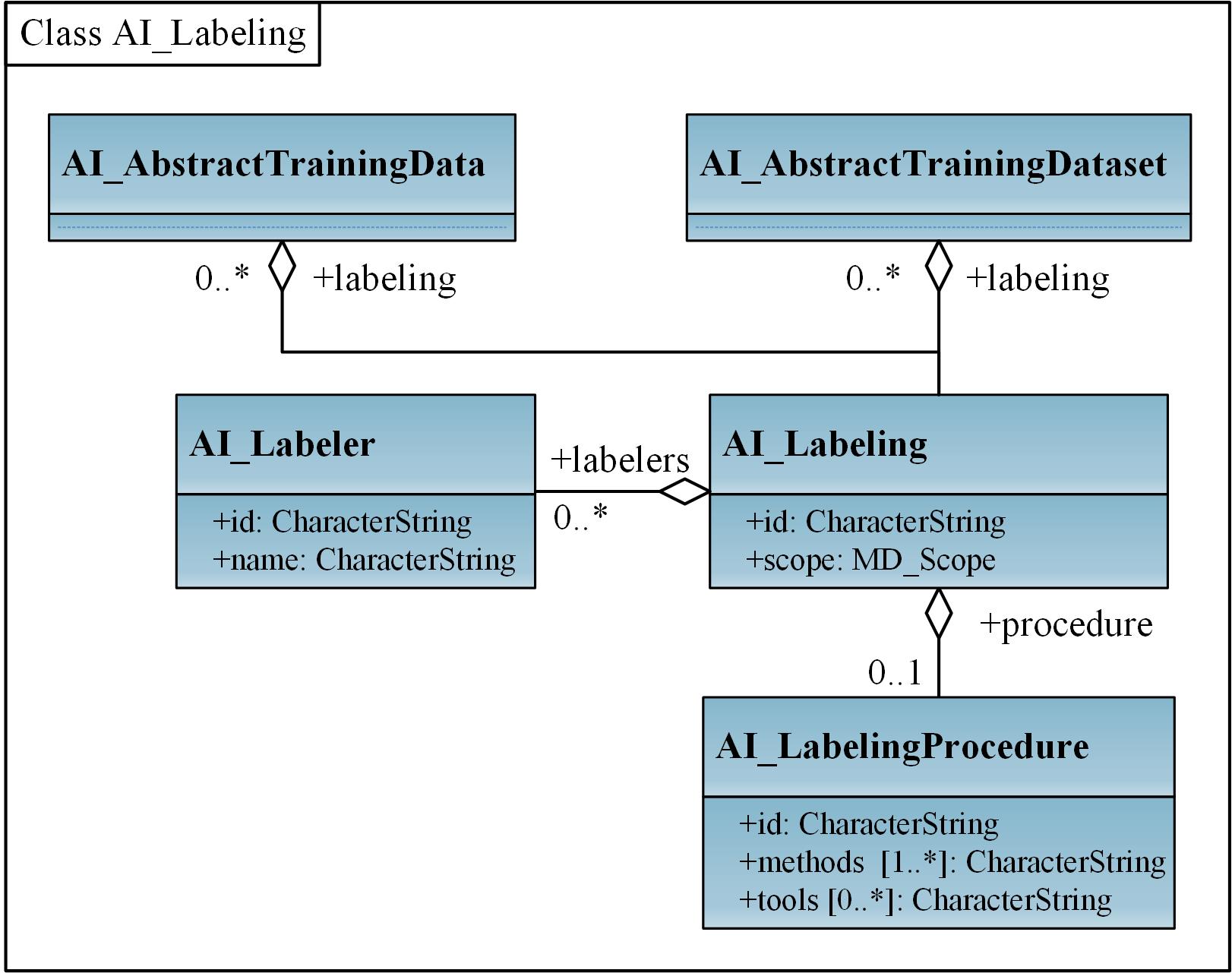


Figure 11. UML diagram of AI\_Labeling

### Requirements

The following requirement defines the rules governing the implementation of the AI\_Labeling conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 5** | |
| /req/ailabeling/classes | |
| For each UML class defined or referenced in AI\_Labeling | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class, including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class, including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 5** | |
| /per/ailabeling/classes | |
| For each UML class defined or referenced in AI\_Labeling | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_Labeling, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 8. Classes defined in AI\_Labeling

|  |  |
| --- | --- |
| Name | Description |
| AI\_Labeling | AI\_Labeling defines basic provenance information on how to create the training dataset and includes the labeler and labeling procedure. |
| AI\_Labeler | AI\_Labeler identifies the agent that creates the labels and can be mapped to the agent in W3C PROV. |
| AI\_LabelingProcedure | AI\_LabelingProcedure represents the process for labeling and can be mapped to the activity in W3C PROV. |

## AI\_TDChangeset

|  |  |
| --- | --- |
| **Requirements class 6** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-aitdchangeset | |
| Target type | Implementation Standard |
| Dependency | /req/req-class-aitrainingdata |

AI\_TDChangeset records changed training samples between two versions in the collection level, including added training samples, modified training samples and deleted training samples. AI\_TDChangeset includes three kinds of changes (add, modify, and delete) for individuals.

The UML diagram of the AI\_TDChangeset is illustrated in Figure 12.

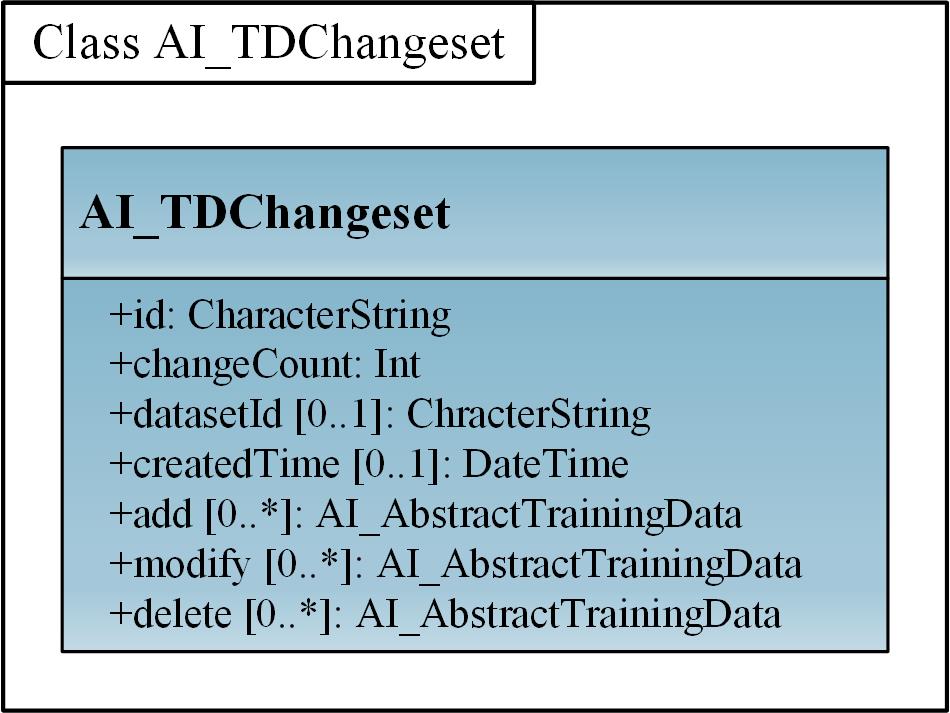


Figure 12. UML diagram of AI\_TDChangeset

### Requirements

The following requirement defines the rules governing the implementation of the AI\_TDChangeset conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 6** | |
| /req/aitdchangeset/classes | |
| For each UML class defined or referenced in AI\_TDChangeset | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class, including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class, including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class, including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 6** | |
| /per/aitdchangeset/classes | |
| For each UML class defined or referenced in AI\_TDChangeset | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the AI\_TDChangeset, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 9. Classes defined in AI\_TDChangeset

|  |  |
| --- | --- |
| Name | Description |
| AI\_TDChangeset | AI\_TDChangeset represents a set of individual level changes (add, modify, and delete) of the training dataset. |

## AI\_DataQuality

|  |  |
| --- | --- |
| **Requirements class 7** | |
| http://www.opengis.net/spec/TrainingDML-AI-1/1.0/req/req-class-aidataquality | |
| Target type | Implementation Standard |
| Dependency | ISO 19157-1 |
| Dependency | /req/req-class-aitrainingdataset |

TD quality description can use DataQuality from ISO 19157-1 to align with the existing efforts on geographic data quality. Data quality can be evaluated in terms of either collection or individual levels of the TD. Although most data quality elements are designed for datasets, it is sometimes needed to know the data quality at the individual level. For example, how is the positional uncertainty of an individual training sample measured by a GPS device in the field? Also in terms of the object label, an individual training sample includes a source image and many object labels. In this case, the quality can be measured based on these labels, such as Completeness. Thus the scope of the data quality can be used to specify the level and extent that identify the data on which data quality is to be established and evaluated. For TrainingDML-AI, AI\_TrainingDataset/AI\_TrainingData can be mapped to dataset/feature in MD\_ScopeCode.

The quality description of the TD can also leverage the quality elements defined in QualityElement. For example, Completeness describes label commission and omission, Thematic accuracy describes class accuracy of labels. TD related quality elements like AI\_ClassBalanceDegree can be defined by extending the QualityElement. AI\_ClassBalanceDegree can help address the bias issue and evaluate whether the number of classes in the training dataset is imbalanced.

The UML diagram of the AI\_DataQuality module is illustrated in Figure 13.

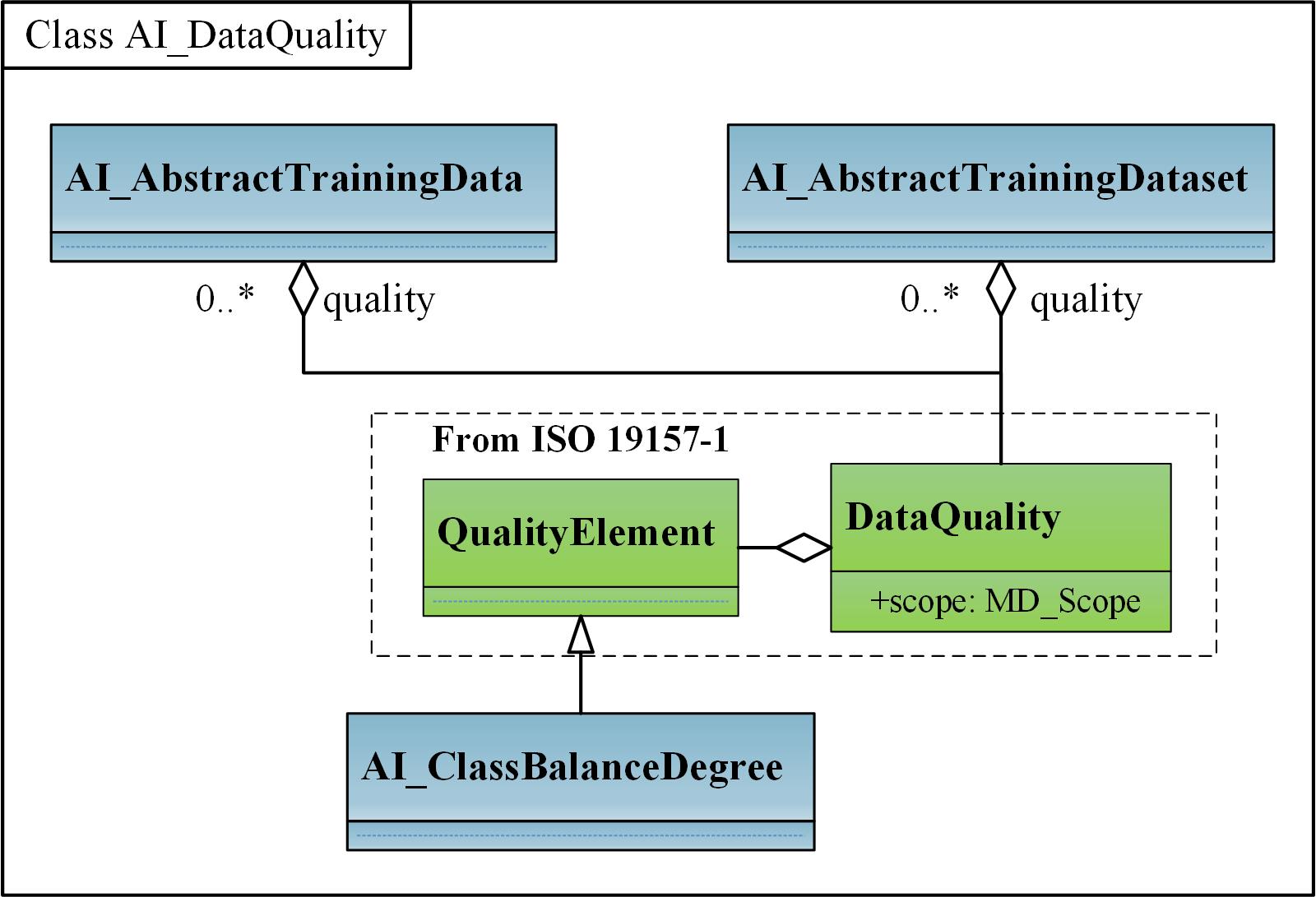


Figure 13. UML diagram of AI\_DataQuality

### Requirements

The following requirement defines the rules governing the implementation of the DataQuality conceptual model as an implementation standard.

|  |  |
| --- | --- |
| **Requirement 7** | |
| /req/aidataquality/classes | |
| For each UML class defined or referenced in DataQuality | |
| A | Any implementation standard SHALL contain an element which represents the same concept as that defined for the UML class. |
| B | Any implementation standard SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class. |
| C | Any implementation standard SHALL represent the attributes of the UML class, including the name, definition, type, and multiplicity. |
| D | Any implementation standard SHALL represent the attributes of all super classes of the UML class, including the name, definition, type, and multiplicity. |
| E | Any implementation standard SHALL represent the associations of all super classes of the UML class, including the source, target, direction, roles, and multiplicity. |
| F | Any implementation standard SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class. |

|  |  |
| --- | --- |
| **Permission 7** | |
| /per/aidataquality/classes | |
| For each UML class defined or referenced in DataQuality | |
| A | An implementation standard MAY represent that class as a null class with no attributes, associations, or definition. |
| B | An implementation standard MAY represent an association of the UML class with a null association. |
| C | An implementation standard MAY represent an attribute of the UML class with a null attribute. |
| D | Whenever a null element is used to represent a concept from the DataQuality, the implementation standard SHOULD document that mapping and provide an explanation for why that concept was not implemented. |

### Class definitions

Table 10. Classes defined in AI\_DataQuality

|  |  |
| --- | --- |
| Name | Description |
| AI\_ClassBalanceDegree | AI\_ClassBalanceDegree defines the quality elements for measuring degree of class balance of training datasets. |

# TrainingDML-AI Data Dictionary

The TrainingDML-AI UML model is the normative definition of the TrainingDML-AI Conceptual Model. The Data Dictionary tables in this section were software generated from the UML model. As such, this section provides a normative representation of the TrainingDML-AI Conceptual Model.

## ISO Classes

The following classes are defined in the ISO standards and used by the TrainingDML-AI Conceptual Model.

### Feature (from ISO 19107:2019)

Table 11. Metadata of Feature (Class)

|  |  |
| --- | --- |
| Definition | Abstraction of real-world phenomena |
| Subclass of | None |
| Stereotype | <<Type>> |

### MD\_Band (from ISO 19115-1:2014)

Table 12. Metadata of MD\_Band (Class)

|  |  |
| --- | --- |
| Definition | MD\_Band is the range of wavelengths in the electromagnetic spectrum. |
| Subclass of | MD\_SampleDimension |
| Stereotype | <<Class>> |

### MD\_Scope (from ISO 19115-1:2014)

Table 13. Metadata of MD\_Scope (DataType)

|  |  |
| --- | --- |
| Definition | The target resource and physical extent for which information is reported. |
| Subclass of | None |
| Stereotype | <<DataType>> |

### EX\_Extent (from ISO 19115-1:2014)

Table 14. Metadata of EX\_Extent (DataType)

|  |  |
| --- | --- |
| Definition | EX\_Extent is the extent of the resource electromagnetic spectrum. |
| Subclass of | None |
| Stereotype | <<DataType>> |

### CI\_Citation (from ISO 19115-1:2014)

Table 15. Metadata of CI\_Citation (Class)

|  |  |
| --- | --- |
| Definition | CI\_Citation is the standardized resource reference. |
| Subclass of | None |
| Stereotype | <<Class>> |

### DataQuality (from ISO 19157-1)

Table 16. Metadata of DataQuality (Class)

|  |  |
| --- | --- |
| Definition | DataQuality is the quality information for the data specified by a data quality scope. |
| Subclass of | None |
| Stereotype | <<Class>> |

### QualityElement (from ISO 19157-1)

Table 17. Metadata of QualityElement (Class)

|  |  |
| --- | --- |
| Definition | QualityElement is the aspect of quantitative quality information. |
| Subclass of | None |
| Stereotype | <<Class>> |

## AI\_TrainingDataset

Table 18. Metadata of AI\_TrainingDataset (ApplicationSchema)

|  |  |
| --- | --- |
| Description | The AI\_TrainingDataset module supports the modelling of different kinds of training dataset. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_AbstractTrainingDataset

Table 19. Metadata of AI\_AbstractTrainingDataset (Class)

|  |  |
| --- | --- |
| Definition | AI\_AbstractTrainingDataset defines the basic concepts and components of different kinds of training dataset. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 20. Attributes of AI\_AbstractTrainingDataset (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identification of the AI training dataset. |
| doi | CharacterString [0..1] | Digital object identifier of the AI training dataset. |
| scope | MD\_Scope [0..1] | Description of the scope of the training dataset. |
| name | CharacterString [1..1] | Name of the AI training dataset. |
| description | CharacterString [1..1] | Description of the AI training dataset. |
| version | CharacterString [0..1] | Version number of the AI training dataset. |
| amountOfTrainingData | Int [1..1] | Total number of training samples in the AI training dataset. |
| createdTime | DateTime [0..1] | Time when the AI training dataset was created. |
| updatedTime | DateTime [0..1] | Time when the AI training dataset was updated. |
| license | CharacterString [0..1] | License description of the AI training dataset. |
| providers | CharacterString [0..\*] | People or organizations who provide the AI training dataset. |
| keywords | CharacterString [0..\*] | Keywords of the AI training dataset. |
| metricsInLIT | AI\_MetricsInLiterature [0..\*] | Results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature. |
| statisticsInfo | NamedValue [0..\*] | Statistics results of training samples in each class. |
| dataSources | CI\_Citation [0..\*] | Citation of data sources. |
| numberOfClasses | Int [1..1] | Total number of classes in the AI training dataset. |
| classificationSchema | CharacterString [0..1] | Classification schema for classes used in the AI training dataset. |
| classes | NamedValue [1..1] | Classes used in the AI training dataset. |

#### AI\_EOTrainingDataset

Table 21. Metadata of AI\_EOTrainingDataset (Class)

|  |  |
| --- | --- |
| Definition | AI\_EOTrainingDataset describes attributes specific to EO training dataset. For example, the data source, image size, band information, and spatial extent. |
| Subclass of | AI\_AbstractTrainingDataset |
| Stereotype | <<Class>> |

Table 22. Attributes of AI\_EOTrainingDataset (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| extent | EX\_Extent [0..1] | Spatial extent of the EO training dataset. |
| bands | MD\_Bands [0..\*] | Description of the image bands used in the EO training dataset. |
| imageSize | CharacterString [0..1] | Size of the images used in the EO training dataset. |

#### AI\_MetricsInLiterature

Table 23. Metadata of AI\_MetricsInLiterature (DataType)

|  |  |
| --- | --- |
| Definition | AI\_MetricsInLiterature records results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature. |
| Subclass of | None |
| Stereotype | <<DataType>> |

Table 24. Attributes of AI\_MetricsInLiterature (DataType)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| doi | CharacterString [1..1] | Digital object identifier of the peer-reviewed literature. |
| algorithm | CharacterString [0..1] | AI/ML algorithms used in the peer-reviewed literature. |
| metrics | NamedValue [1..\*] | Metrics and results of AI/ML algorithms in the peer-reviewed literature. |

## AI\_TrainingData

Table 25. Metadata of AI\_TrainingData (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_TrainingData module supports the modelling of an individual AI training sample. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_AbstractTrainingData

Table 26. Metadata of AI\_AbstractTrainingData (Class)

|  |  |
| --- | --- |
| Definition | AI\_AbstractTrainingData defines the basic concepts and components of different kinds of training data. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 27. Attributes of AI\_AbstractTrainingData (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identification of the individual AI training sample. |
| datasetId | CharacterString [0..1] | Identification of the training dataset that the training sample belongs to. |
| trainingType | AI\_TrainingTypeCode [0..1] | Training type of the individual AI training sample. |
| numberOfLabels | Int [0..1] | Total number of labels in the individual AI training sample. |
| dataSources | CI\_Citation [0..\*] | Citation of inputs to prepare a training sample. |

#### AI\_EOTrainingData

Table 28. Metadata of AI\_EOTrainingData (Class)

|  |  |
| --- | --- |
| Definition | AI\_EOTrainingData describes attributes specific to EO training data. For example, spatial extent and date time of the source EO inputs. |
| Subclass of | AI\_AbstractTrainingData |
| Stereotype | <<Class>> |

Table 29. Attributes of AI\_EOTrainingData (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| extent | EX\_Extent [0..1] | Spatial extent of the individual EO training sample. |
| dateTime | DateTime [0..\*] | Data times when the EO data were obtained. |
| dataURL | URL [1..\*] | URLs of the EO data. |

#### AI\_TrainingTypeCode

Table 30. Metadata of AI\_TrainingTypeCode (CodeList)

|  |  |
| --- | --- |
| Definition | AI\_TrainingTypeCode is a code list used to identify whether the individual is used for different purposes. |
| Subclass of | None |
| Stereotype | <<CodeList>> |

## AI\_Task

Table 31. Metadata of AI\_Task (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_Task module supports the modelling of different kinds of AI task. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_AbstractTask

Table 32. Metadata of AI\_AbstractTask (Class)

|  |  |
| --- | --- |
| Definition | AI\_AbstractTask defines the AI/ML tasks in specific domains that have different organizing forms of TD. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 33. Attributes of AI\_AbstractTask (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identification of the task. |
| datasetId | CharacterString [0..1] | Identification of the training dataset the training sample belongs to. |
| description | CharacterString [0..1] | Description of the AI task. |

#### AI\_EOTask

Table 34. Metadata of AI\_EOTask (Class)

|  |  |
| --- | --- |
| Definition | AI\_EOTask extends AI\_AbstractTask to represent specific AI/ML tasks in the EO domain. |
| Subclass of | AI\_AbstractTask |
| Stereotype | <<Class>> |

Table 35. Attributes of AI\_EOTask (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| taskType | CharacterString [1..1] | Type description of the EO task. |

## AI\_Label

Table 36. Metadata of AI\_Label (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_Label module supports the modelling of different kinds of AI Label. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_AbstractLabel

Table 37. Metadata of AI\_AbstractLabel (Class)

|  |  |
| --- | --- |
| Definition | AI\_AbstractLabel defines a set of informative tags or metadata obtained by labelling training data. AI\_AbstractLabel can be represented using a feature, coverage, or a semantic class from ontologies or shared vocabularies. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 38. Attributes of AI\_AbstractLabel (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| isNegative | Bool [0..1] | Whether the training sample related to the label is a positive or negative sample. |
| confidence | Float [0..1] | Confidence score of the labeler. |

#### AI\_SceneLabel

Table 39. Metadata of AI\_SceneLabel (Class)

|  |  |
| --- | --- |
| Definition | AI\_SceneLabel represents the scene level label using a semantic type from ontologies or shared vocabularies. |
| Subclass of | AI\_AbstractLabel |
| Stereotype | <<Class>> |

Table 40. Attributes of AI\_SceneLabel (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| class | CharacterString [1..1] | Class name that records the semantic type of the scene of the training sample. |

#### AI\_ObjectLabel

Table 41. Metadata of AI\_ObjectLabel (Class)

|  |  |
| --- | --- |
| Definition | AI\_ObjectLabel represents the object level label using a feature. |
| Subclass of | AI\_AbstractLabel |
| Stereotype | <<Class>> |

Table 42. Attributes of AI\_ObjectLabel (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| object | Feature [1..1] | Feature that represents the position and attributes of the object. |
| bboxType | CharacterString [0..1] | Type of the bbox. |
| class | CharacterString [1..1] | Class that records the semantic of the object type. |
| isDiffDetectable | Bool [0..1] | Whether the object is difficult to detect. |
| dateTime | DateTime [0..1] | Created time of the object label. |

#### AI\_PixelLabel

Table 43. Metadata of AI\_PixelLabel (Class)

|  |  |
| --- | --- |
| Definition | AI\_PixelLabel represents the pixel level label using a coverage. |
| Subclass of | AI\_AbstractLabel |
| Stereotype | <<Class>> |

Table 44. Attributes of AI\_PixelLabel (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| imageURL | URL [1..\*] | URL of the images representing the label information. |
| imageFormat | CharacterString [1..1] | Image data format. |

## AI\_Labeling

Table 45. Metadata of AI\_Labeling (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_Labeling module supports the modelling of provenance information of the training dataset. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_Labeling

Table 46. Metadata of AI\_Labeling (Class)

|  |  |
| --- | --- |
| Definition | AI\_Labeling defines basic provenance information on how to create the training dataset. AI\_Labeling includes the labeler and labeling procedure. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 47. Attributes of Labeling (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identifier of the labeling. |

#### AI\_Labeler

Table 48. Metadata of AI\_Labeler (Class)

|  |  |
| --- | --- |
| Definition | AI\_Labeler identifies the agent that creates the labels and can be mapped to the agent in W3C PROV. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 49. Attributes of AI\_Labeler (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identifier of the labeler. |
| name | CharacterString [1..1] | Name of the labeler. |

#### AI\_LabelingProcedure

Table 50. Metadata of AI\_LabelingProcedure (Class)

|  |  |
| --- | --- |
| Definition | AI\_LabelingProcedure represents the process for labeling and can be mapped to the activity in W3C PROV. |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 51. Attributes of AI\_LabelingProcedure (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identifier of the labeling procedure. |
| methods | CharacterString [1..\*] | Methods used in the labeling procedure. |
| tools | CharacterString [0..1] | Tools or software used in the labeling procedure. |

## AI\_TDChangeset

Table 52. Metadata of AI\_TDChangeset (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_TDChangeset module supports the modelling of change information of the training dataset. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_TDChangeset

Table 53. Metadata of AI\_TDChangeset (Class)

|  |  |
| --- | --- |
| Definition | AI\_TDChangeset |
| Subclass of | None |
| Stereotype | <<Class>> |

Table 54. Attributes of AI\_TDChangeset (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| id | CharacterString [1..1] | Identifier of the changeset. |
| datasetId | CharacterString [0..1] | Identifier of the training dataset the changeset belongs to. |
| version | CharacterString [0..1] | Version of the training dataset that the changeset belongs to. |
| changeCount | Int [1..1] | Total number of changed training samples. |
| createdTime | DateTime [0..1] | Created time of the changeset. |
| add | AI\_AbstractTrainingData [0..\*] | Added training samples. |
| modify | AI\_AbstractTrainingData [0..\*] | Modified training samples. |
| delete | AI\_AbstractTrainingData [0..\*] | Deleted training samples. |

## AI\_DataQuality

Table 55. Metadata of AI\_DataQuality (ApplicationSchema)

|  |  |
| --- | --- |
| Description | AI\_DataQuality module supports the modelling of quality description of different kinds of training datasets. |
| Parent package | TrainingDML-AI |
| Stereotype | <<ApplicationSchema>> |

### Classes

#### AI\_ClassBalanceDegree

Table 56. Metadata of AI\_ClassBalanceDegree (Class)

|  |  |
| --- | --- |
| Definition | AI\_ClassBalanceDegree defines the quality elements for measuring degree of class balance of training datasets. |
| Subclass of | QualityElement |
| Stereotype | <<Class>> |

Table 57. Attributes of AI\_ClassBalanceDegree (Class)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value type and multiplicity** | **Definition** |
| measure | MeasureReference [1..1] | Reference to measure used |
| evaluationMethod | EvaluationMethod [1..1] | Evaluation information |
| result | QualityResult [1..\*] | Value obtained from applying a data quality measure |

1. Abstract Test Suite (Normative)
   * 1. Introduction

The test method for TrainingDML-AI conceptual model specified in this ATS is manual inspection. Automated methods may be used where they exist.

Conformance class AI\_TrainingDataset

|  |  |
| --- | --- |
| **Abstract test A.1** | |
| /ats/aitrainingdataset/classes | |
| Requirement | /req/aitrainingdataset/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_TrainingDataset Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_TrainingData

|  |  |
| --- | --- |
| **Abstract test A.2** | |
| /ats/aitrainingdata/classes | |
| Requirement | /req/aitrainingdata/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_TrainingData Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_Task

|  |  |
| --- | --- |
| **Abstract test A.3** | |
| /ats/aitask/classes | |
| Requirement | /req/aitask/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_Task Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_Label

|  |  |
| --- | --- |
| **Abstract test A.4** | |
| /ats/ailabel/classes | |
| Requirement | /req/ailabel/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_Label Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_Labeling

|  |  |
| --- | --- |
| **Abstract test A.5** | |
| /ats/ailabeling/classes | |
| Requirement | /req/ailabeling/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_Labeling Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_TDChangeset

|  |  |
| --- | --- |
| **Abstract test A.6** | |
| /ats/aitdchangeset/classes | |
| Requirement | /req/aitdchangeset/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_TDChangeset Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

Conformance class AI\_DataQuality

|  |  |
| --- | --- |
| **Abstract test A.7** | |
| /ats/aidataquality/classes | |
| Requirement | /req/aidataquality/classes |
| Test purpose | To validate that the implementation standard correctly implements the UML Classes defined in the Conceptual Model. |
| Test method type | Manual Inspection |
| Test method | 1. For each UML class defined or referenced in the AI\_DataQuality Package:    1. Validate that the implementation standard contains a data element which represents the same concept as that defined for the UML class.    2. Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicity as those documented in the Conceptual Model.    3. Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    4. Validate that the properties of the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.    5. Validate that the associations represented for the data element include those of all super classes of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model.    6. Validate that the implementation standard enforces all constraints imposed on the UML class by the Conceptual Model. |

1. Example (Informative)

TrainingDataset encoding examples

WHU-RS19 dataset

The WHU-RS19 dataset is widely used in scene classification of remote sensing images. This dataset is collected from Google Earth and has 19 classes including airport, beach, bridge, commercial, desert, farmland, football field, forest, industrial, meadow, mountain, park, parking, pond, port, railway station, residential, river, and viaduct. Each class contains around 50 images, with the image size 600×600 and a resolution of 0.5m.

An example of JSON encoding of the WHU-RS19 dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/WHU-RS19.json>.

DOTA-v1.5 dataset

The DOTA-v1.5 dataset is a large-scale dataset for object detection in aerial images. The sources for content in the dataset include Google Earth, Gaofen-2, and Jilin-1 imagery provided by China Resources Satellite Data Center. The 16 classes in DOTA-v1.5 are plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field, swimming pool, and container crane. Compared with other aerial image object detection datasets, the dataset has the largest number of classes. The images in the dataset have various image sizes (from 800×800 to 2000×2000) and resolutions (Google Earth/0.1m-1m, Gaofen-2/1m, Jilin-1/0.72m).

An example of JSON encoding of the DOTA-v1.5 dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/DOTA-v1.5.json>.

KITTI 2D object detection dataset

The KITTI 2D object detection dataset is a novel open-access dataset and benchmark for road area and ego-lane detection. KITTI 2D consists of 7481 annotated training images of high variability from the KITTI autonomous driving platform by 2 PointGrey Flea2 color cameras, capturing a broad spectrum of urban street views and road scenes. The eight (8) classes in the KITTI 2D object detection dataset are car, van, truck, pedestrian, person\_sitting, cyclist, tram, and misc. Compared with other street view object detection datasets, this dataset compresses diverse scenarios and captures real-world traffic situations, ranging from freeways over rural areas to inner-city scenes with many static and dynamic objects.

An example of JSON encoding of the KITTI 2D object detection dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/KITTI.json>.

GID dataset

The GID dataset is one of start-of-art land cover classification datasets. This dataset has a large spatial coverage covering many provinces in China with a relatively high spatial resolution (2m). GID has two sets. One is the GID-5C. It has 150 images (image size 7200×6800) that are classified into 5 land cover classes. The other set is GID-15C. The images from GID-5C are sliced into 30,000 patches in GID-15C, which have three types of patch sizes (56×56, 112×112, 224×224) and are classified into 15 land cover classes.

An example of JSON encoding of the GID-5C dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/GID-5C.json>.

Toronto3D dataset

The Toronto3D dataset is a large urban outdoor point cloud dataset for segmentation collected by the Mobile Laser Scanning System. The dataset covers about 1 km of scene streets in Toronto, including four areas named L001, L002, L003, and L004, with a total of 78.3 million points. Each point in this dataset has 10 attributes representing the 3D position, RGB color, intensity, GPS time, scan angle rank, and category, respectively. This dataset has eight categories, including road, road mark, natural, building, utility line, pole, car, and fence.

An example of JSON encoding of the Toronto3D dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/Toronto_3D.json>.

WHU-Building dataset

The WHU-Building dataset is a change detection dataset collected from the Land Information New Zealand Data Service. The dataset is composed of images (with the resolution 0.2m) in 2012 and 2016, covering 20.5 km2. It includes 12,796 and 16,077 buildings respectively in 2012 and 2016.

An example of JSON encoding of the WHU-Building dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/WHU-building.json>.

California change detection dataset

The California Change Detection Dataset is composed of two images and a label image. The first image is a Landsat 8 acquisition covering Sacramento County, Yuba County and Sutter County, California, on 5 January 2017. It has nine channels covering the spectrum from deep blue to short-wave infrared, plus two long-wave infrared channels. The second image was acquired on 18 February 2017 by Sentinel-1A over the same area after the occurrence of a flood. The image is recorded in polarizations VV and VH and augmented with the ratio between the two intensities as a third channel. All these channels are log-transformed.

An example of JSON encoding of the California change detection dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/UiT_HCD_California_2017.json>.

WHU MVS dataset

The WHU MVS dataset is a synthetic aerial dataset created for large-scale and high-resolution Earth surface reconstruction. The basic training sample of the dataset is a multi-view unit consisting of five aerial images, and their corresponding depth maps are taken as ground truth. There are a total of 5680 pairs of five-view aerial images in the dataset. All the images are simulated from a 3D surface model, which is produced by Smart3D software using Unmanned Aerial Vehicle (UAV) images and refined by manual editing.

An example of JSON encoding of the WHU MVS dataset following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/WHU_MVS.json>.

DataQuality encoding example

WHU-RS19 data quality

An encoded data quality example of the WHU-RS19 datasets following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/WHU-RS19-quality.json>.

TDChangeset encoding example

DOTA-v1.5 changeset

DOTA-v1.5 uses the same images as DOTA-v1.0, but the extremely small instances (less than 10 pixels) are also annotated. Moreover, a new category “container crane” is added. It contains 403,318 instances in total. The number of images and dataset splits are the same as DOTA-v1.0. This version was released for the DOAI Challenge 2019 on Object Detection in Aerial Images in conjunction with IEEE CVPR 2019.

An encoded changeset example between the DOTA-v1.0 and DOTA-v1.5 datasets following the TrainingDML-AI UML model can be found in <https://gitlab.ogc.org/ogc/TrainingDML-AI/-/blob/master/use-cases/examples/DOTA-v1.5-changeset.json>.

1. Revision history (Informative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Author | Paragraph modified | Description |
| 2022-08-01 | 0.1 | Peng Yue, Boyi Shangguan | all | Draft for internal review |
| 2022-09-09 | 0.2 | Peng Yue, Boyi Shangguan, Carl Reed | all | Merge edits and comments from Carl Reed. |
| 2022-11-10 | 0.3 | Peng Yue, Boyi Shangguan | all | Revision based on comments from Testbed-18 TrainingDataset. |
| 2023-01-13 | 0.8 | Peng Yue, Boyi Shangguan | all | Revision based on comments from Testbed-18 ML TD ER. |
| 2023-01-29 | 1.0 | Peng Yue, Boyi Shangguan, Ruixiang Liu | all | Internal review done and submission to request OAB review. |

1. Bibliography

[1] ISO, 2019. ISO 19107: 2019. Geographic information — Spatial schema. <https://www.iso.org/standard/26012.html>

[2] ISO, 2022. ISO 19157-1: 2022. Geographic information — Data quality. <https://www.iso.org/standard/32575.html>

[3] ISO, 2014. 19115-1:2014, Geographic information — Metadata — Part 1: Fundamentals. <https://www.iso.org/standard/53798.html>

[4] Landry, T., ed., 2018. OGC Testbed-14: Machine Learning Engineering Report, OGC 18-038r2. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/18-038r2.html>

[5] Meek, S., ed., 2019. OGC Testbed-15: Machine Learning Engineering Report, OGC 19-027r2. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/19-027r2.html>

[6] Schumann, G., ed., 2020. OGC Testbed-16: Machine Learning Training Data Engineering Report, OGC 20-018. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/20-015r2.html>

[7] Yue, P., Shangguan, B., Hu, L., Jiang, L., Zhang, C., Cao, Z., Pan, Y., 2022. Towards a training data model for artificial intelligence in earth observation. International Journal of Geographical Information Science, 1-25. <https://doi.org/10.1080/13658816.2022.2087223>

1. [www.opengeospatial.org/cite](http://www.opengeospatial.org/cite) [↑](#footnote-ref-1)