|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ITU logo | INTERNATIONAL TELECOMMUNICATION UNION  **TELECOMMUNICATION STANDARDIZATION SECTOR**  STUDY PERIOD 2017-2020 | | | SG13-TD297/PLEN | |
| **STUDY GROUP 13** | |
| **Original: English** | |
| **Question(s):** | | | 17/13 | Virtual, 20-31 July 2020 | |
| **TD** | | | | | |
| **Source:** | | | Editors | | |
| **Title:** | | | Draft new Recommendation ITU-T Y.3531 (formerly Y.MLaaS-reqts): “Cloud computing- Functional requirements for machine learning as a service” – for consent | | |
| **Purpose:** | | | Decision | | |
| **Contact:** | | Sungpil Shin ETRI Korea (Republic of) | | | Tel: +82-42-860-1379 Fax: +82-42-861-5404 Email: [spshin@etri.re.kr](mailto:spshin@etri.re.kr) |
| **Contact:** | | Xiaowu HE China Telecom People’s Republic of China | | | Tel: +86 20 38639580 Fax: +86 20 38636215 [hexw4.gd@chinatelecom.cn](mailto:hexw4.gd@chinatelecom.cn) |

|  |  |
| --- | --- |
| **Keywords:** | machine learning, machine learning as a service; cloud computing |
| **Abstract:** | This document contains the proposed draft Recommendation ITU-T Y.3531(formerly, MLaaS-reqts): “Cloud computing - Functional requirements of machine learning as a service” for consent. |

Draft new Recommendation ITU-T Y.3531 (formerly Y.MLaaS-reqts)

Cloud computing – Functional requirements for machine learning as a service

Summary

This Recommendation provides cloud computing requirements for machine learning as a service (MLaaS), which addresses requirements from use cases. Machine learning as a service is a cloud service category in which the capability provided to the cloud service customer is the provision and use of machine learning framework. Machine learning framework is a set of functionalities for provisioning machine learning data as well as training, deploying, and managing machine learning model.

On the perspective of cloud computing service provisioning, this Recommendation provides the functional requirements for MLaaS to identify functionalities such as machine learning data pre-processing, machine learning model training, machine learning model testing, and etc. Also, this Recommendation aligned with the cloud computing reference architecture of [ITU-T Y.3502].

Keywords

MLaaS; machine learning; machine learning as a service; cloud computing; big data; big data as a service; BDaaS

**CONTENTS**

| Page |
| --- |
| [1 Scope 4](#_Toc46931620)  [2 References 4](#_Toc46931621)  [3 Definitions 4](#_Toc46931622)  [3.1 Terms defined elsewhere 4](#_Toc46931623)  [3.2 Terms defined in this Recommendation 5](#_Toc46931624)  [4 Abbreviations and acronyms 6](#_Toc46931625)  [5 Conventions 6](#_Toc46931626)  [6 Overview of machine learning 6](#_Toc46931627)  [6.1 Introduction to machine learning 6](#_Toc46931628)  [6.2 Generic process of machine learning 7](#_Toc46931629)  [6.3 Machine learning ecosystem 9](#_Toc46931630)  [7 Machine Learning as a Service 11](#_Toc46931637)  [7.1 System context of MLaaS 11](#_Toc46931638)  [7.2 CSN:machine learning data provider 12](#_Toc46931639)  [7.3 CSN:machine learning model developer 13](#_Toc46931640)  [7.4 CSP:machine learning service provider 13](#_Toc46931641)  [7.5 CSC:machine learning service user 15](#_Toc46931642)  [8 Functional requirements of MLaaS 15](#_Toc46931643)  [9 Security considerations 19](#_Toc46931644)  [Appendix I Use case of MLaaS for operation perspectives 21](#_Toc46931645)  [Appendix II Use case of MLaaS for application perspectives 31](#_Toc46931654)  [Bibliography 40](#_Toc46931661) |

Draft new Recommendation ITU-T Y.3531 (formerly Y.MLaaS-reqts)

Cloud computing – Functional requirements for machine learning as a service

# Scope

This Recommendation provides system context, functional requirements, and use cases for machine learning as a service (MLaaS).

In particular, the scope of this Recommendation includes:

* Overview of machine learning;
* Introduction to MLaaS;
* Functional requirements of MLaaS.

The use cases of MLaaS are developed to derive functional requirements of MLaaS.

NOTE – Developments of machine learning algorithms and methodologies are out of the scope on this Recommendation.

# References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below.

[ITU-T Y.3500] Recommendation ITU-T Y.3500 (2014), *Information technology* – *Cloud computing* – *Overview and vocabulary.*

[ITU-T Y.3501] Recommendation ITU-T Y.3501 (2013), *Cloud computing framework and high-level requirements.*

[ITU-T Y.3502] Recommendation ITU-T Y.3502 (2014), *Information technology* – *Cloud computing* – *Reference architecture.*

[ITU-T Y.3510] Recommendation ITU-T Y.3510 (2013), *Cloud computing infrastructure requirements*.

[ITU-T Y.3600] Recommendation ITU-T Y.3600 (2015), *Big data – Cloud computing based requirements and capabilities*.

[ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*

# Definitions

## Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

* + 1. **activity** [ITU-T Y.3502]:A specified pursuit or set of tasks.
    2. **cloud computing** [ITU-T Y.3500]:Paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand.

NOTE – Examples of resources include servers, operating systems, networks, software, applications and storage equipment.

* + 1. **cloud service** [ITU-T Y.3500]: One or morecapabilities offered viacloud computing(3.1.2) invoked using a defined interface.
    2. **cloud service customer** [ITU-T Y.3500]: Party which is in a business relationship for the purpose of using cloud services.

NOTE – A business relationship does not necessarily imply financial agreements.

* + 1. **cloud service partner** [ITU-T Y.3500]:Party which is engaged in support of, or auxiliary to, activities of either the cloud service provider or the cloud service customer, or both.
    2. **cloud service provider** [ITU-T Y.3500]:Party which makes cloud services available.
    3. **machine learning** [ITU-T Y.3172]: Processes that enable computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed.

NOTE 1 – This definition is adapted from [b-ETSI GR ENI 004].

NOTE 2 – Supervised machine learning and unsupervised machine learning are two examples of machine learning types.

* + 1. **machine learning model** [ITU-T Y.3172]: Model created by applying machine learning techniques to data to learn from.

NOTE 1 – A machine learning model is used to generate predictions (e.g., regression, classification, clustering) on new (untrained) data.

NOTE 2 – A machine learning model may be encapsulated in a deployable fashion in the form of a software (e.g., virtual machine, container) or hardware component (e.g., IoT device).

NOTE 3 – Machine learning techniques include learning algorithms (e.g., learning the function that maps input data attributes to output data).

* + 1. **metadata** [b-ITU-T H.752]:Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment and management of the described entities.
    2. **role** [ITU-T Y.3502]:A set of activities that serves a common purpose.
    3. **sub-role** [ITU-T Y.3502]:A subset of the activities of a given role.

## Terms defined in this Recommendation

This Recommendation defines the following terms:

* + 1. **machine learning as a service (MLaaS)**: A cloud service category in which the capabilities provided to the cloud service customer is the provision and use of machine learning framework.
    2. **machine learning framework:** A set of functionalities for provisioning machine learning data, as well as training, deploying, and managing machine learning model.

# Abbreviations and acronyms

This Recommendation uses the following abbreviations:

|  |  |
| --- | --- |
| CSC | Cloud Service Customer |
| CSN | Cloud Service Partner |
| CSP  ML  MLaaS  MLDP  MLMD  MLSP  MLSU | Cloud Service Provider  Machine Learning  Machine Learning as a Service  Machine Learning Data Provider  Machine Learning Model Developer  Machine Learning Service Provider  Machine Learning Service User |

# Conventions

The following conventions are used in this Recommendation:

* The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.
* The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.
* The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.

# Overview of machine learning

## Introduction to machine learning

Machine learning (ML) is a technique in the fields of computer science to enable machines or computers to learn how to perform tasks without being explicitly programmed. According to the definition, the mechanism of ML is distinct from explicit programming which applies if-then statements for making decisions. The goal of machine learning is to improve their ability to solve the tasks automatically through the data.

The development of ML considers ML models, learning algorithms, and training data. The ML model is defined as mathematical representations for performing the tasks with input data. And the learning algorithm works to train the ML model with the training data for the task.

The ML developer configures the ML model and learning algorithm, and gathers training data based on setting up the task. The task are designed to solve the problems with evaluation metrics for the performance based on computational methods. The evaluation metric is criteria for measuring the performance of ML model. The examples of general task for ML are prediction, classification, clustering, generating samples, and etc.

The ML tasks are classified depending on the characteristics of discipline such as necessity for training data or feedback. The categories of ML tasks include supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

* **Supervised learning**: a task of learning with labelled data which map an input data to a desired output data.
* **Unsupervised learning**: a task of learning with unlabelled data, which finds structures or patterns in the data.
* **Semi-supervised learning**: a task of learning with both labelled and unlabelled data, which is in the situations of small amount of labelled data.
* **Reinforcement learning**: a task of learning to achieve the agent to perform the task with the environment data and feedback.

## Generic process of machine learning

The process is implemented with performing ML data acquisition, ML data processing, ML model development, and ML model deployment. Figure 6-1 shows the generic process of ML.

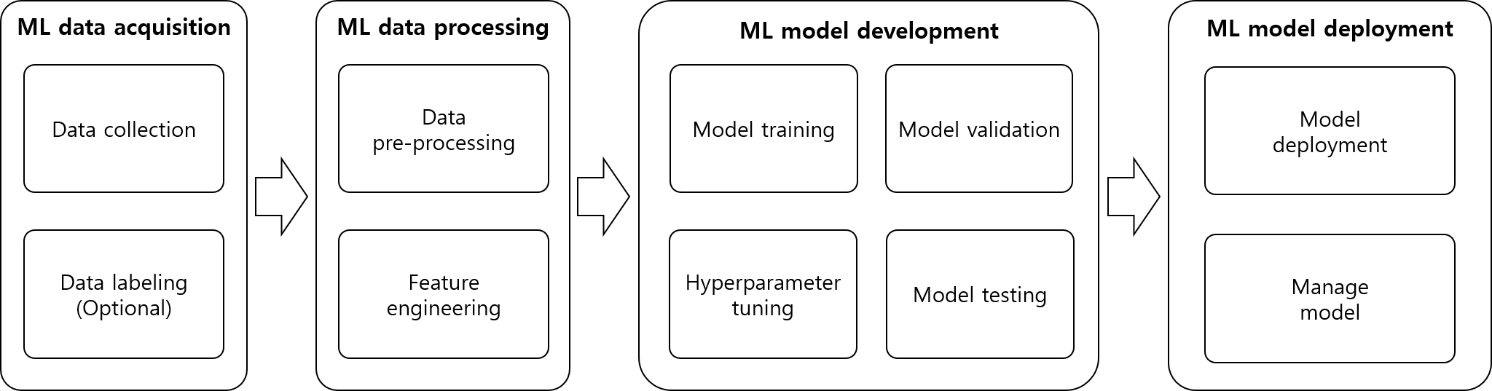


Figure 6-1 – Generic process of machine learning

* **ML data acquisition**: is to collect the data for training. The collected data are grouped into training dataset, validation dataset, and test dataset before ML model development .
  + **Data collection**: is to gather the raw data including structured data, unstructured data, and semi-structured data. Within all data types data can exist in formats, such as text, spreadsheet, video, audio, image, map, etc. [ITU-T Y.3600].
  + **Data labelling**: is optionally performed for generating labelled data for supervised learning or semi-supervised learning. Human resources and the labelling tools are required for making annotations of data. The tools have unit for tagging, for example, image data use tagging unit of bounding box to indicate objective, and video data use tagging unit of 5 second videos for the labelling.
* **ML data processing**: is handling the data for improving learning performance or creating meaningful information from the data.
  + **Data pre-processing**: is transforming the data to resolve inaccurate, incomplete, or unreasonable of the data. The noisy and bias in the raw data are removed or mitigated for training data.
  + **Feature engineering**: is finding and determining the useful features for ML model from the data. Feature engineering algorithms includes feature selection, scaling, and extraction to implement ML.
    - **Feature selection**: is obtaining a subset of the features from the original features with same or similar analytical results. The examples of feature selections are genetic algorithm, greedy forward selection, correlation feature selection, and etc.
    - **Feature scaling**: is normalizing the range of data in the original features. The examples of feature scaling are mean normalization, min-max normalization, and etc.
    - **Feature extraction**: is deriving new features from the original features by transforming into a reduced set of features. The examples of feature extractions are principal component analysis, dimensionality reduction, and etc.
* **ML model development**: is the process for training and optimizing the ML model. The model training, model validation, model testing, and hyperparameter tuning are iteratively performed to meet the purpose of ML model.
  + **Model training:** is to fit the parameters of ML model with the training dataset. Training algorithms are implemented for update the parameters such as backpropagation, gradient descent, and etc. The parameters are adjusted or optimized automatically by feeding training dataset.
  + **Model validation:** is verifying the performance of ML model with validation dataset. The model validation process gives opportunity to optimize the hyperparameters before the ML training is completed.
  + **Model testing:** is measuring the performance of final trained ML model with test dataset. Model testing give the final performance report for deploying the trained ML model. The Model testing prevents the re-learning when the ML model shows poor performance after ML model is deployed.
  + **Hyperparameter tuning:** is adjusting the hyperparameters of ML model training. The hyperparameters are adjustable values for controlling ML model training. The hyperparameters include iteration time, batch size, epoch, and etc.
* **ML model deployment**: is utilizing the model for performing the task in applications. The ML model deployment involves deploying and managing model including retraining model, monitoring the performance of model, managing model, and etc.
  + **Model deployment**: is to load the trained ML model in the application or hardware. ML developer use the trained model for developing ML applications.
  + **Manage models**: is to update or monitor the ML model. In addition, manage models involves requesting retraining model as well.

Figure 6-2 shows an example of developing trained ML model which includes the life cycle following the generic process of ML.

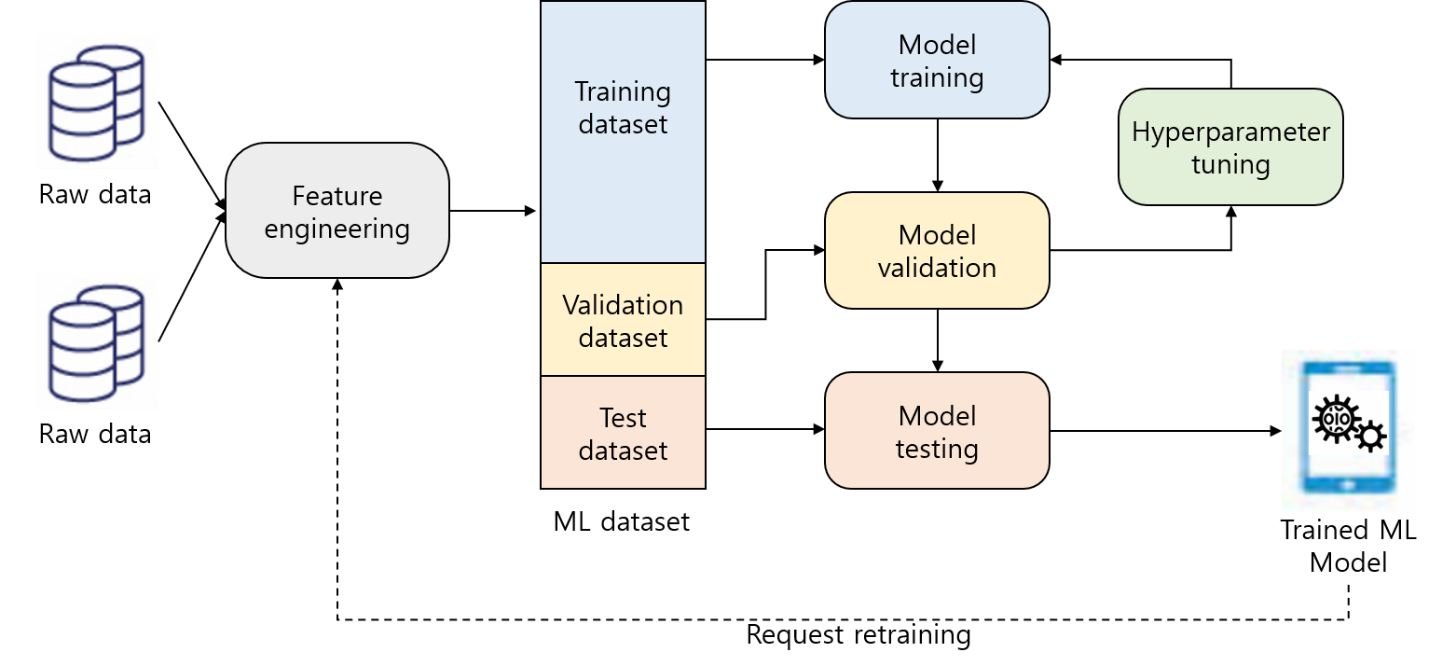


Figure 6-2 – Example of generic ML process

In Figure 6-2, the final output of machine learning process is trained ML model which have the ability to perform the tasks in the devices. If a deployed trained ML model falls below an acceptable performance, then the model needs to be retrained or reengineered. The deployed model is retrained on new dataset to meet the required performance on the purpose.

For achieving the goal of ML, the ML requires well-designed models and learning algorithms, as well as the amount of data for training. The several learning algorithms and ML models are developed to improve the performance for solving the tasks.

## Machine learning ecosystem

There is an ecosystem in the fields of ML which participate stakeholder’s related with providing ML data, ML model, and ML framework. In this ecosystem, the developers can effectively build their own ML or application.

This clause describes a machine learning ecosystem through stakeholder’s roles and sub-roles. It defines necessary activities for roles providing and consuming machine learning as well as relationships between roles.

The machine learning ecosystem includes the following roles:

* Data provider;
* ML model provider;
* ML framework provider
* ML framework customer.

The machine learning ecosystem is shown in Figure 6-3.

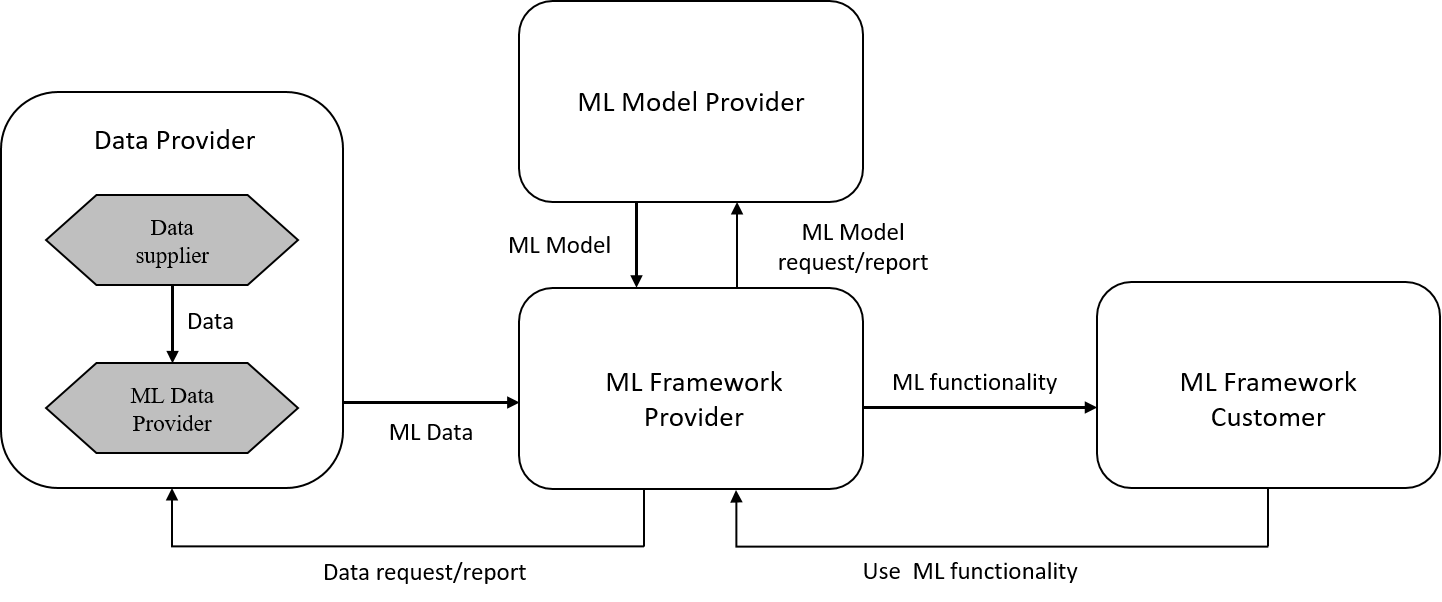


Figure 6-3 – High-level machine learning ecosystem

### Data provider

The data provider (DP) role consists of two sub-roles:

* Data supplier;
* ML data provider.

### Data supplier

The data supplier provides data from different sources, and performs same activities which is defined in [ITU-T Y.3600]. The data supplier's activities include:

* Generating data;
* Creating metadata information describing the data source(s) and relevant attributes;
* Publishing metadata information to access the metadata.

### ML data provider

The ML data provider acquires data and performs data processing for machine learning processing. It receives data from data supplier [ITU-T Y.3600] then sends data for ML processing to ML framework provider. The ML data provider supports various kinds of data, e.g. structured data, unstructured data, streaming data, etc.

The ML data provider's activities include:

* Analysing features from data;
* Generating data set for ML.

NOTE 1 – The activities such as analysing data and pre-processing are described in [ITU‑T Y.3600].

NOTE 2 – Analysing features from data includes labelling task with human resources. This activity can be performed optionally if ML model requires labelled data for training.

### ML model provider

The ML model provider provides ML model to ML framework provider. Developing ML model involves ML algorithms for handling data, learning models, and evaluating models in the process, as well as designing the schema of ML model. The ML model provider performs developing and providing ML model.

The ML model provider's activities include:

* Developing ML model;
* Providing ML model.

### ML framework provider

The ML framework provider provides ML functionalities to ML framework customer. The ML framework includes interface for using ML functionalities. The ML framework provider also performs managing and reporting activities to ML framework customer for developing ML model.

The ML framework provider's activities include:

* Training ML model with ML data;
* Managing and reporting ML model;
* Deploying ML model for ML application;
* Providing ML functionalities to ML framework customer.

### ML framework customer

The ML framework customer uses ML functionalities from ML framework provider for business, e.g., decision making, business process automation and customer interaction, etc. The ML framework can be end-user or a system that performs ML-enabled applications.

The ML framework customer's activities include:

* Accessing and utilizing ML framework.

# Machine Learning as a Service

## System context of MLaaS

Machine learning as a service (MLaaS) is a cloud service category in which the capabilities provided to the cloud service customer (CSC) is the provision and use of machine learning framework. For the processing of ML, it needs a large amount of computing power and resources for ML model training due to a large amount of training data and the high complex computation of ML model training. MLaaS provides the benefit to resolve the problem by providing elastic computing capabilities and resources in the cloud environments based on CSC’s requests.

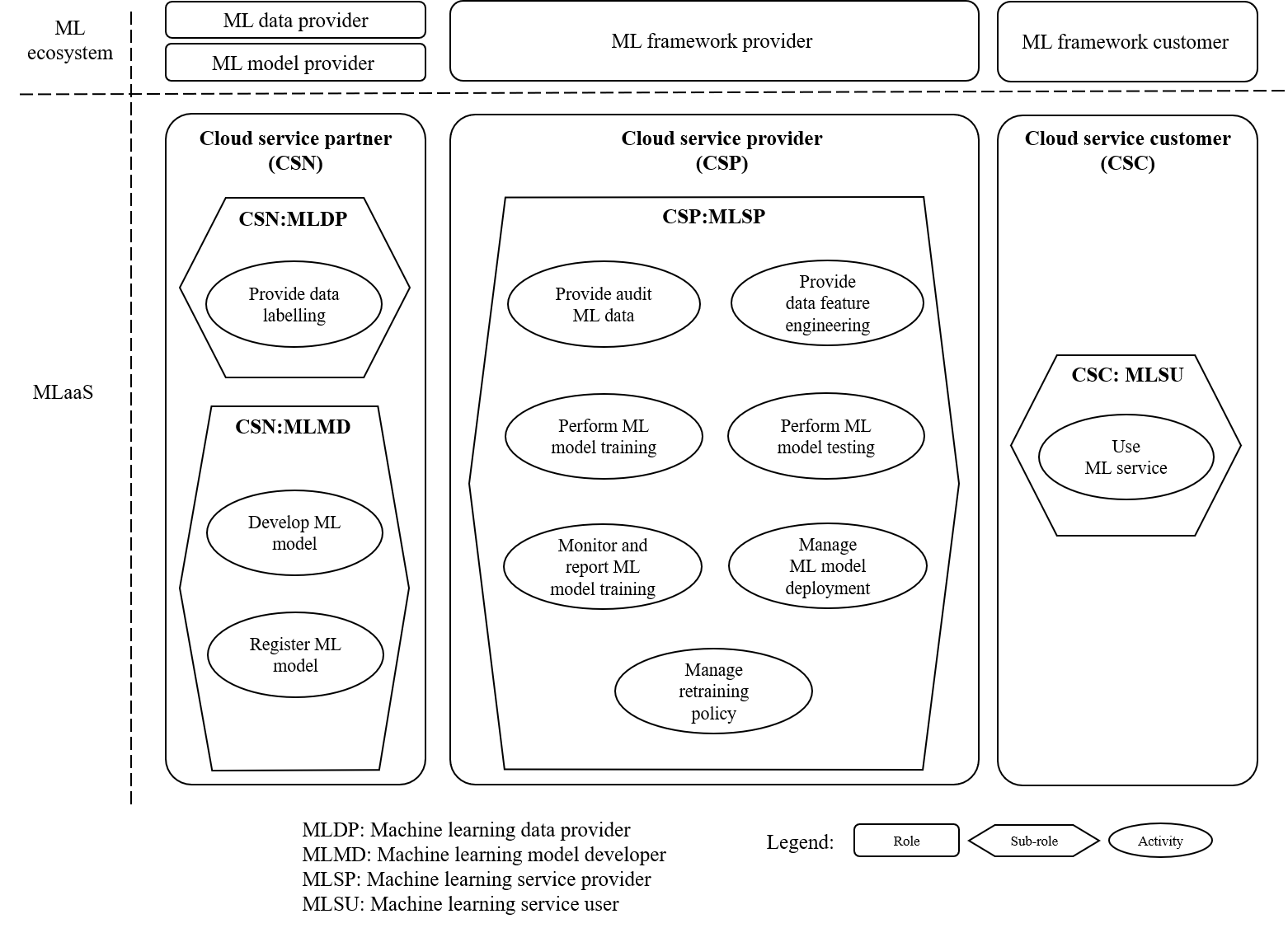
System context of MLaaS provides additional sub-roles and activities based on the architectural user view defined in [ITU-T Y.3502]. This clause describes how cloud computing supports the four main roles in a machine learning ecosystem: ML data provider, ML model provider, ML framework provider, and ML framework customer.

Cloud computing sub-roles can be mapped to machine learning roles as shown in Table 7-1. The sub-roles of cloud service provider (CSP), the cloud service partner (CSN), and CSC mapped with the roles and sub-roles of machine learning ecosystems.

**Table 7-1 – Mapping table between machine learning ecosystem and   
MLaaS System context**

|  |  |
| --- | --- |
| **Machine learning ecosystem** | **MLaaS System context** |
| ML data provider | CSN:MLDP(machine learning data provider) |
| ML model provider | CSN:MLMD(machine learning model developer) |
| ML framework provider | CSP:MLSP(machine learning service provider) |
| ML framework customer | CSC:MLSU(machine learning service user) |

Figure 7-1 illustrates the cloud computing sub-roles for MLaaS. Figure 7-1 also identifies activities specific for machine learning and assigns them to cloud computing sub-roles. MLaaS utilizes other sub-roles of CSP and CSN.



**Figure 7-1 – MLaaS system context**

## CSN:machine learning data provider

The CSN:MLDP is a sub-role of the CSN, which provides data labelling and CSN:DP’s activities in [ITU-T Y.3600]. The activities of CSN:DP are generate data, publish data, and brokerage data. The additional activities of CSN:MLDP include:

– provide data labelling.

**7.2.1 Provide data labelling**

The provide data labelling activity involves generating the labelling information with the tools for labelling task. The human labellers manually generate the annotation for the data with given guideline in the tools. This activity involves:

* generating labelling with guidelines from the CSP:MLSP;
* providing metadata of labelled data;
* updating a catalogue of CSN:DP for CSP:MLSP to search ML data.

## CSN:machine learning model developer

The CSN:MLMD is a sub-role of the CSN, which supports ML model to solve various learning task. The CSN:MLMD generate a ML model catalogue for discovery and utilization of ML model. ML model catalogue include the metadata of ML model such as scheme of ML model, version of ML model, usage of ML model, and evaluation metric for ML model.

NOTE – The ML model usage include the applicable learning task of ML model.

The CSN:MLMD’s activities include:

* develop ML model;
* register ML model.

**7.3.1 Develop ML model**

The develop ML model activity involve developing and updating machine learning models, and publishing ML model with the information of ML model. This activity involves:

* developing ML model with setting initial values of parameter;
* updating ML model with the feedback, and reporting information;
* providing configurations of hyperparameters for ML model training;
* generating metadata of ML model including its usage, input/output data format, expected performance, and etc.

**7.3.2 Register ML model**

The register ML model activity is the process of providing a ML model catalogue to CSP:MLSP. The ML model catalogue including the information of ML model with metadata.

NOTE – The ML model can be provided with structured format defined by ML frameworks.

This activity involves:

* providing an access information of the ML model to the CSP:MLSP;
* providing a catalogue to the CSP:MLSP for searching appropriate ML model.

## CSP:machine learning service provider

The CSP:MLSP is a sub-role of the CSP, which provides MLaaS service including infrastructures and tools for training, deploying, and managing ML model. In addition, the CSP:MLSP supports to collect and audit ML data.

NOTE – The CSP:MLSP can provide the CSP:BDIP’s activities in [ITU-T Y.3600]. The related activities of CSP:BDIP are performing data collection, performing data storage, providing data pre-processing, and providing data integration.

The additional activities of CSP:MLSP include:

* Provide audit ML data;
* Provide data feature engineering;
* Perform ML model training;
* Perform ML model testing;
* Monitor and report ML model training;
* Manage ML model deployment;
* Manage retraining policy.

**7.4.1 Provide audit ML data**

The provide audit ML data activity supports labelling tasks with providing universalized and managed tools for labelling task. This activity encourages to improving labelling quality and reliability of the ML data set. This activity involves:

* providing labelling environments and tools for the CSN:MLDP;
* auditing ML data for labelling quality;
* reporting a feedback to the CSN:MLDP.

**7.4.2 Provide data feature engineering**

The provide data feature engineering activity performs data feature engineering methods such as feature selection, feature scaling, and feature extraction form ML data. The feature engineering supports to speed up the training process and manage the risk of underperforming ML model.

NOTE – The overfitting problem is an example of underperforming ML model. The overfitting occurs when the model fits the data too much, and fails to generalize for prediction.

**7.4.3 Perform ML model training**

The perform ML model training activity is executing training process for ML model including validation of ML model. This activity involves:

* creating virtual machine and storage for ML model training;
* triggering and operating ML model training with input for training relevant information such as configuration of hyperparameter, and etc;
* providing configurations for partitioning the ML data into training dataset, validation dataset and test dataset;
* validating performance of the ML model;
* storing and registering the results of ML model training.

**7.4.4 Perform ML model testing**

The perform ML model testing activity is evaluating trained ML model before it deployed. This activity ensures the performance and quality of trained ML model for target the tasks.

**7.4.5 Monitor and report ML model training**

The monitor and report ML model training activity supports model training process by providing functionalities such as measuring resource utilization, alerting abnormalities, and automatic stopping. The history of report can be utilized for detecting the error of ML model scheme and determining retraining policy. This activity involves:

* measuring and monitoring resource utilizations during ML model training;
* reporting a problem if ML model training cannot be maintained with provided ML model;
* providing the performance reports of ML model during ML model training;
* executing the automatic stopping if the configured threshold is exceeded.

NOTE – The CSC:MLSU configures threshold values for automatic stopping such as resource utilization overloads, and etc.

**7.4.6 Manage ML model deployment**

The manage ML model deployment activity supports to export the trained ML model for deploying into target computing environment. The CSP:MLSP supports to transform multiple format of ML model for the various computing environments. In addition, the exported models are registered for retraining and updating ML model catalogue.

NOTE 1 – The trained ML model can be updated in catalogue and provided for servicing ML model for transfer learning.

NOTE 2 – The transfer learning is technique to use pre-trained ML model for applying related learning task.

**7.4.7 Manage retraining policy**

The manage retraining policy activity is defining the process of retraining for the CSC:MLSU. The retraining is implemented for ensuring and maintaining the performance of ML model or improving the performance of ML model. The retraining policy considers the expected performance of a deployed model, monitoring time period, the following action, and etc.

## CSC:machine learning service user

The CSC:MLSU is a sub-role of the CSC, which utilizes ML framework and cloud services to develop ML applications according to the user’s intention.

The CSC:MLSU’s activity includes:

* use ML service.

**7.5.1 Use ML service**

The use machine learning service activity involves invocating and using the machine learning framework and cloud service for developing ML applications.

# Functional requirements of MLaaS

* 1. **ML data collection and storage requirements**

[ITU-T Y.3600] provides data collection and storage functional requirements in terms of big data. The functionalities of MLaaS also involved data collection and data storage, and several functional requirements of ML data collection and storage are not different with data collection and storage functional requirements of big data in [ITU-T Y.3600].

The following functional requirements are inherited from big data functional requirements in [ITU-T Y.3600] with changing the role from big data to machine learning such as CSP:BDIP to CSP:MLSP and CSN:DP to CSN:MLDP.

The data collection and storage requirements include:

1) It is required for the CSP:MLSP to support collecting data from multiple CSN:MLDPs in parallel;

2) It is recommended for the CSN:MLDP to expose data to the CSP:MLSP by publishing metadata;

3) It is recommended that the CSP:MLSP supports collecting data from different CSN:MLDPs with different modes;

NOTE 1 – Data could be collected in different modes, such as pull mode in which the data collection process is initiated by CSP:MLSP, or push mode in which the data collection process is initiated by the CSN:MLDP.

4) It is recommended for the CSN:MLDP to provide a brokerage service to the CSP:MLSP for searching accessible data;

NOTE 2 – Brokerage provides data a catalogue which has data information such as data specification, data instructions, electronic access methods, license policy, data quality, etc.

5) It is required for the CSP:MLSP to support different data types with sufficient storage space, elastic storage capacity and efficient control methods;

6) It is required for the CSP:MLSP to support storage for different data formats and data models;

NOTE 3 – Data formats include text, spreadsheet, video, audio, image, map, etc. Data models include relational models, document models, key-value models, graph models, etc. (as described in clause 6.1 in [ITU-T Y.3600]).

* 1. **ML data labelling requirements**

The ML data labelling requirements include:

1) It is required that CSP:MLSP provide tools of labelling tasks for CSN:MLMD.

NOTE 1 – The tools of labelling tasks provides different type of tagging unit for the data format such as image, video, text, and etc.

2) It is required that CSP:MLSP provide audit data for labelled ML data.

NOTE 2 – The audit data for labelling are utilized to enhance data labelling quality such as accuracy of labelling, consistency of labelling and etc.

NOTE 3 – The audit data is reported to CSN:MLDP as a feedback when the request arrived or when the labelling quality is poor for developing ML model.

3) It is required that CSN:MLDP provides information of data labelling task.

NOTE 4 – The information of data labelling task include data type, unit of labelling, consensus majorities of labellers, and etc.

* 1. **ML data pre-processing requirements**

With the same manner in clause 8.1, the following requirements are partially inherited from big data functional requirements in [ITU-T Y.3600] with changing the role from big data to machine learning such as CSP:BDIP to CSP:MLSP.

The ML data pre-processing requirements include:

1) It is required for the CSP:MLSP to support data aggregation.

NOTE 1 – Data from different sources can be organized in the same model or data format, as described in clause 6.1 of [ITU-T Y.3600].

2) It is recommended that the CSP:MLSP supports unification of data collected in different formats;

NOTE 2 – Unification of data is for example to unify data about persons/locations/dates extracted from web pages, pictures, videos, SNS data and calling logs to text format.

3) It is recommended for the CSP:MLSP to support extraction of data from unstructured data or semi-structured data into structured data.

NOTE 3 – This requirement can be applied also to data storage.

The additional requirements of ML data pre-processing not described in [ITU-T Y.3600] include:

4) It is required that the CSP:MLSP to provide configuration of splitting the ML data into training dataset, validation dataset and test dataset.

NOTE 4 – The training dataset, validation dataset, and test data set are divided with completely independent.

* 1. **ML data analysis and feature engineering requirements**

With the same manner in clause 8.1, the following requirements are inherited from big data functional requirements in [ITU-T Y.3600] with changing the role from big data to machine learning such as CSP:BDIP to CSP:MLSP.

The ML data analysis requirements include:

1) It is required for the CSP:MLSP to support analysis of various data types and formats;

2) It is required for the CSP:MLSP to support association analysis;

NOTE 1 – Association analysis is the task of uncovering relationships among data.

3) It is required for the CSP:MLSP to support different data analysis algorithms;

NOTE 2 – Data analysis algorithms include classification, clustering, regression, association, ranking, etc.

The additional requirements of ML data feature engineering not described in [ITU-T Y.3600] include:

4) It is recommended that the CSP:MLSP provide feature selection for determining a subset of relevant features of ML data.

5) It is recommended that the CSP:MLSP provide feature scaling for normalizing the range of features of ML data.

6) It is recommended that the CSP:MLSP provide feature extraction for generating new improved features from the original features of ML data.

* 1. **ML model training requirements**

The ML model training requirements include:

1) It is required that CSN:MLMD provide registry of the ML model and catalogue to CSP:MLSP.

2) It is recommended that CSP:MLSP provide feedback on ML model usage to CSP:MLMD.

NOTE 1 – The feedback includes applied learning task with high performance experienced by CSC:MLSU. The CSP:MLMD use the feedback for updating the ML model.

3) It is required that CSP:MLSP provide configuration the hyperparameters value.

4) It is recommended that CSN:MLMD provide default configuration values of hyperparameters.

5) It is recommended that CSN:MLMD provide the restricted range of values for hyperparameter adjustments.

NOTE 2 – The restricted range of values are the adoptable values of hyperparameter for the ML model.

6) It is recommended that CSP:MLSP provide visualization of learning results.

NOTE 3 – The examples of visualization are providing chart, table, distribution plot, and graph which shows the analytic information of learning result.

7) It is required that CSP:MLSP provides the operations of ML model training.

NOTE 4 – The operations of ML model training include initiate, stop, and resume of ML model training .

8) It is required that the CSP:MLSP provide validation process for the ML models with validation dataset.

9) It is recommended that CSP:MLSP provide monitoring the learning status during ML model training.

NOTE 5 – The learning status includes expected learning time, applied hyperparameter set, memory usage, and etc.

10) It is required that CSP:MLSP provides performance evaluation for a trained ML model.

11) It is required that CSP:MLSP stores evaluation results of trained ML models with applied hyperparameters set.

12) It is recommended that CSP:MLSP provides hyperparameter optimization methods.

NOTE 6 – The hyperparameter optimization is choosing a set of optimal hyperparameters for a ML model training. The example of hyperparameter optimization methods are Grid search, Bayesian optimization, and evolutionary optimization.

13) It is recommended that CSP:MLSP provides automated ML model search methods.

NOTE 7 – The automated ML model search is finding the design of ML model in a way to increasing the performance for the target learning task.

14) It is recommended that CSP:MLSP provide a transformation of ML models for use in other ML framework.

15) It is recommended that CSP:MLSP provide distributed ML model training.

NOTE 8 – The distributed ML model training is the training of ML model in multiple worker nodes for accelerating.

* 1. **ML model monitoring requirements**

The ML model monitoring requirements include:

1) It is required that CSP:MLSP provide monitoring of the resource utilization during ML model training.  
NOTE 1 – The resource utilization includes processing unit (such as CPU/GPU), memory, storage, and network utilization.

2) It is recommended that CSP:MLSP alert the resource utilization overloads during ML model training.

3) It is required that CSP:MLSP reports the history of the resource utilization with timestamps.

4) It is required that CSP:MLSP provide the automatic stopping by detecting learning failure or measuring unpromising performance of model.

NOTE 2 – The detection of learning failure includes ML parameter update failures.

5) It is recommended that CSP:MLSP provide setting the values of threshold for the automatic stopping.

NOTE 3 – The automatic stopping is executed with the user defined values of threshold for avoiding unwanted results in early. The unwanted results include overtraining, decreasing performance, and etc.

6) It is recommended that CSN:MLMD provide the default values of threshold for automatic stopping.

7) It is required that CSP:MLSP store the history for learning failure during ML model training.

NOTE 4 – The history includes log of the resource utilization, the performance measurement, and execution of automatic stopping.

* 1. **Trained ML model deploying and retraining requirements**

The trained ML model deploying and retraining requirements include:

1) It is required that the CSP:MLSP provide the registry of the trained ML model.

NOTE 1 – The trained ML model is registered with schema including applied input/output data, the structures of ML model, and etc.

2) It is required that the CSP:MLSP provide metadata of the trained ML models.

NOTE 2 – The metadata of trained ML model includes the evaluated performance, applied hyperparameter set, and etc.

3) It is required that the CSP:MLSP export the trained ML model with applicable format for deploying target hardware.

4) CSP:MLSP can optionally provide monitoring of the performance for the deployed ML model.

5) It is required that CSP:MLSP provide retraining policy for managing the performance of trained ML model.

NOTE 3 – The retraining policy includes reset learning parameter, add new data for learning for optimizing the ML model.

6) It is recommended that the CSP:MLSP provide ML model retraining according to measured performance of the ML model.

# Security considerations

It is recommended that the security framework for cloud computing described in [b-ITU-T X.1601] be considered for the machine learning as a service. [b-ITU-T X.1601] analyses security threats and challenges in the cloud computing environment and describes security capabilities that could mitigate these threats and meet security challenges.

[b-ITU-T X.1631] provides guidelines supporting the implementation of information security controls for cloud service customers and cloud service providers. Many of the guidelines guide the cloud service providers to assist the cloud service customers in implementing the controls and guide the cloud service customers to implement such controls. Selection of appropriate information security controls and the application of the implementation guidance provided, will depend on a risk assessment as well as any legal, contractual, regulatory or other cloud-sector specific information security requirements.

Relevant security requirements of [b-ITU-T Y.2201], [b-ITU-T Y.2701] and applicable X, Y and M series of ITU-T Recommendations need to be taken into consideration, including access control, authentication, data confidentiality, data retention policy, network security, data integrity, availability and privacy.

Appendix I  
Use case of MLaaS for operation perspectives   
  
(This appendix does not form an integral part of this Recommendation.)

The use cases in the Appendix I provide the examples of operating MLaaS functionalities and related functional requirements of MLaaS.

## I.1 ML data annotation/labelling management

|  |  |
| --- | --- |
| Title | ML data annotation/labelling management |
| Description | This use case describes procedure of managing ML data. The management procedure includes assigning data, generating annotation, reporting result, and merging ML data. The following are specific steps for managing ML data with annotators.   1. CSP:MLSP set the ML data server and collect the raw data from CSN:DP. 2. CSP:MLSP request the annotation work with assigning raw data. 3. CSN:MLDP perform annotation work with provided annotating method from CSP:MLSP. 4. CSN:MLDP provided the annotated data set to CSP:MLSP. 5. CSP:MLSP perform decision policy for the annotated data.    1. CSP:MLSP decide ‘accept’ or ‘reject’ for annotated data from each annotator.    2. CSP:MLSP save/reports the results of quality of annotated data.    3. CSP:MLSP merge the accepted data into the ML data set. |
| Role/Sub-role | CSN:MLDP |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSN:MLDP operates ML data server for scheduling/assigning data resources to annotators.  CSN:MLDP searches and request raw data for annotation from CSN:DP. |
| Post-conditions  (optional) | CSN:MLDP provides the interface for reporting ML result and learning progress to CSC:MLSU.  CSN:MLSP stores the trained/optimized ML model for developing ML applications. |
| Derived requirements | – Clause 8.2 requirement (1,2,3)  – Clause 8.1 requirement (1,3,5,6) |

## I.2 Model training with user configuration

|  |  |
| --- | --- |
| Title | Model training with user configuration |
| Description | This use case describes procedure of model training in cloud computing. The single machine for training is considered as a default training. The following is general step for the model training in this use case.   1. CSP:MLSP installs virtual machine with machine learning engine which has interface for model training to user. 2. CSN:MLMD provide ML model for ML training to CSP:MLSP. 3. CSN:MLMD provide appropriate ML data for ML model to CSP:MLSP.   NOTE – For the preparation of ML model and data pair, CSC:MLSU request the ML model and appropriate ML data for model, or CSP:MLSP provide pair of ML model an ML data for CSC:MLSU. This preparation scenario is beyond the scope of this use case.   1. CSC:MLSU configure and set the learning policy and learning parameter for machine learning training. 2. CSP:MLSP performs to train ML model and track the performance of training result for reporting to CSC:MLSU. 3. CSP:MLSP save the trained model and training result into the designated source. |
| Role/Sub-role | CSN:MLMD  CSN:MLDP  CSP:MLSP  CSC:MLSU |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSC:MLSU installs virtual machine with CSP for building ML model.  CSP provides ML framework/platform tools for building ML model.  CSN:MLSU searches and request ML data and model from CSN:MLDP and CSN:MLMD. |
| Post-conditions  (optional) | CSP:MLSP provides the interface for reporting ML result and learning progress to CSC:MLSU.  CSP:MLSP stores the trained/optimized ML model for developing ML applications. |
| Derived requirements | – Clause 8.1 requirement (1,2,4)  – Clause 8.3 requirement (1,2,3,4)  – Clause 8.5 requirement (1,3,6,7,9,10,11) |

## I.3 Report learning result and re-training ML model

|  |  |
| --- | --- |
| Title | Report learning result and re-training ML model |
| Description | This use case describes report learning result from CSP:MLSP to CSC:MLSU, and re-training ML model. The objective of reporting learning result is generally to give information for handling and managing ML training configuration for CSC:MLSU. CSC:MLSU can optimize ML learning parameter and modify the ML learning policy by hand with these reported information, as well as CSC:MLSU can request re-training option. The following is general step for this use case.   1. CSC:MLSU requests learning report to CSP:MLSP. 2. CSP:MLSP transforms and visualizes the learning report to appropriate interfaces.   NOTE 1 – The visualization options can be provided to CSC:MLSU from CSP:MLSP. The raw data of the training result is default option.   1. CSP:MLSP reports the learning result to CSC:MLSU. 2. CSC:MLSU analyzes the learning report and manages and optimizes the learning policies.   NOTE 2 – Reset & re-training steps are the same with the training procedure in ‘model training with user configuration’ use case. |
| Role/Sub-role | CSP:MLSP  CSC:MLSU |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSN:MLSP already performs ML learning and stores backup result data for ML model. |
| Post-conditions  (optional) |  |
| Derived requirements | – Clause 8.6 requirement (1,2,3,7)  – Clause 8.7 requirement (5) |

## I.4 Distributed training with multiple worker nodes

|  |  |
| --- | --- |
| Title | Distributed training with multiple worker nodes |
| Description | This use case describes model training with multiple worker nodes for supporting parallel and distributed learning. CSC:MLSU can organize the multiple worker nodes for ML training. This option has many advantages such as reductions of data size for each worker node and can divide the data for private and public. The following is general step for the distributed training.   1. CSC:MLSU designs/organizes the architecture of ML worker nodes. 2. CSP:MLSP provide the ML management for distributed training to CSC:MLSU. 3. CSC:MLSU set the distribution policy for ML model/data and scheduling policy for assigning resources and ML parameters. 4. CSP:MLSP performs resource and parameter assigning with the configured policy from CSC:MLSU. 5. CSP:MLSP request ML training to each virtual/local worker node and collect the training results. 6. CSP:MLSP iterates the 4) and 5) steps until training is completed. |
| Role/Sub-role | CSP:MLSP  CSC:MLSU |
| Figure (optional) | Figure 1 Allocation of learning data to multiple worker nodes    Figure 2 Distributed learning and updating parameter to ML management |
| Pre-conditions  (optional) | CSP provides virtual server for managing ML parameters and control distributed learning policy.  CSP provides network connection with multiple worker nodes on other cloud and local environment. |
| Post-conditions  (optional) |  |
| Derived requirements | – Clause 8.5 requirement (3,6,7,8,9,10,15) |

## I.5 Model testing and optimizing the model quality includes hyperparameter tuning

|  |  |
| --- | --- |
| Title | Model testing and optimizing the model quality includes hyperparameter tuning |
| Description | This use case describes procedure of model testing in cloud computing. The model testing or validation process is usually performed for optimizing or generalizing model’s performance and quality. The testing may require iterative process with tuning some model perspective resources and hyperparameter. Generally, the poor performance of ML model causes from 1) lack of feature prediction, 2) nonoptimal hyperparameter, and 3) abnormal learning data. The purpose of the model testing usually related with resolving the problems of 1) and 2). In that context, this use case of model testing mainly target on resolving 1) and 2) problems experimentally.  The following is general step for the model testing in this use case.   1. CSN:MLMD requests learning result to CSP:MLSP. 2. CSP:MLSP reports the initial/previous learning result to CSN:MLMD.   NOTE – The Model testing process can be performed iteratively.   1. CSN:MLMD optimizes the ML model with given data set.    1. CSN:MLMD tunes the hyperparameter of ML model.    2. CSN:MLMD adds the feature for ML data. 2. CSN:MLMD tests the ML model until the performance of ML model is qualified. |
| Role/Sub-role | CSN:MLMD  CSP:MLSP  CSC:MLSU  CSN:MLDP |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSP:MLSP installs virtual machine and builds ML model for testing.  CSP provides ML framework/platform tools for testing ML model. |
| Post-conditions  (optional) | CSN:MLDP provides the ML data set for learning and testing.  CSP:MLSP stores the testing history of ML model for debugging ML.  If CSN:MLMD fails to optimize the ML model, then CSN:MLMD requests testing ML data to CSN:MLDP. |
| Derived requirements | – Clause 8.5 requirement (2,3,4,5,6,12) |

## I.6 Model monitoring for alerting abnormal or unsuspected learning process

|  |  |
| --- | --- |
| Title | Model monitoring for alerting abnormal or unsuspected learning process |
| Description | This use case describes the function for monitoring services in machine learning as a service. The abnormalities during learning may be caused from many aspects which can end up the failure of learning, so the CSC:MLSU or CSN:MLMD should be able to recognize the abnormal state of learning by monitoring their systems. The abnormalities during the learning process can be detected through resource overload in the hardware such as CPU, and GPU, excessively abnormal prediction results, synchronization error of parameters, and others.  The following is general step for the model testing in this use case.   1. CSC:MLSU or CSN:MLMD requests job details for learning process. 2. CSP:MLSP reports the job details which shows current job statuses. NOTE – The job statuses include CPU or GPU utilization, memory usage, parameter update/synchronization status, and etc. 3. CSC:MLSU requests to stop the learning procedure to CSP:MLSP.   Also the CSN:MLMD can set the automatic stop when the learning abnormality is clearly detected. In that case, CSN:MLMD can set the values which are served by CSP:MLSP or customized by CSN:MLMD. |
| Role/Sub-role | CSN:MLMD  CSP:MLSP  CSC:MLSU  CSN:MLDP |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSP:MLSP installs virtual machine and builds ML model for testing. |
| Post-conditions  (optional) | CSP:MLSP provides the interface for reporting the testing result to CSN:MLMD.  CSN:MLDP provides the ML data set for learning and testing.  CSP:MLSP stores the testing history of ML model for debugging ML.  If CSN:MLMD fails to optimize the ML model, then CSN:MLMD requests testing ML data to CSN:MLDP. |
| Derived requirements | – Clause 8.5 requirement (10)  – Clause 8.6 requirement (1,2,3,4,5,6,7) |

## I.7 Model deploying and monitoring

|  |  |
| --- | --- |
| Title | Model deploying and monitoring |
| Description | This use case describes procedures of model deploying and monitoring in cloud computing environment. Once a model has been trained, then it is deployed into production. Model monitoring process is also needed to maintain the performance of deployed model. In a broad sense, model deploying and monitoring in this use case includes registering, managing and monitoring stages in cloud computing environment.  The following are general steps for model deploying and monitoring described in this use case.   1. CSP:MLSP registers a trained model after training stage. 2. CSP:MLSP provides information about the trained model used by CSP:MLSP for developing an application service. 3. CSP:MLSP provides an application service using the deployed model, and monitor continuously its performance. 4. CSP:MLSP asks model retraining or reengineering according to measured performance of the model and predefined policy. 5. CSP:MLSP retrains or reengineers a model when being asked. 6. CSP:MLSP notifies CSP:MLSP using the model for an application service when it is retrained or reengineered. 7. CSC:MLSU uses the application service using the deployed model. |
| Role/Sub-role | CSP:MLSP  CSC:MLSU |
| Figure (optional) | The following figures show model deploying and monitoring procedures.    NOTE – Model monitoring is a stage which monitors continuously the performance of a deployed model and determine a following process, e.g. retraining or reengineering the model, if needed. |
| Pre-conditions  (optional) | CSP:MLSP provides a trained model to be deployed for developing an application service. |
| Post-conditions  (optional) | CSC:MLSU uses an ML application service provided by CSP:MLSP. |
| Derived requirements | – Clause 8.7 requirement (1,2,3,4,5,6) |

## I.8 Automated machine learning in cloud computing

|  |  |
| --- | --- |
| Title | Automated machine learning in cloud computing |
| Description | This use case describes procedures of automated machine learning in cloud computing environment. The automated machine learning supports three main functionalities of algorithms which are automated feature engineering, ML model search, and hyperparameter optimization. For executing automated machine learning, CSC:MLSU just configures the learning task and input data. Then, the automated feature engineering algorithms are constructing features of ML data for ML model training automatically. After feature constructing, the ML model search algorithms are implemented for finding a design of ML model by exploring the ML model catalogue. Finally, the hyperparameter optimization algorithms are tuning the values of hyperparameters to maximize the performance of ML model.  The following are general steps for the model deployment in this use case.   1. CSC:MLSU set the learning task and input data. 2. CSP:MLSP performs feature engineering algorithms with input data. 3. CSP:MLSP performs ML model search algorithms and hyperparameter optimization algorithms iteratively.   NOTE 1 – The ML model search algorithms and hyperparameter optimization algorithms can be performed by combining both algorithms  NOTE 2 – The validation of ML model is performed during the step 3).   1. CSP:MLSP tests the ML model which is output of 3). 4-1) If test performance is evaluated under target performance, then repeat the step 3). 4-2) Otherwise, exports the trained ML model. |
| Role/Sub-role | CSP:MLSP  CSC:MLSU |
| Figure (optional) | The following figures show workflow of automated machine learning.    Figure – Workflow of automated machine learning |
| Pre-conditions  (optional) |  |
| Post-conditions  (optional) |  |
| Derived requirements | – Clause 8.5 requirement (12,13,14)  – Clause 8.6 requirement (4,5,6) |

Appendix II  
Use case of MLaaS for application perspectives   
  
(This appendix does not form an integral part of this Recommendation.)

The use cases in the Appendix II provide the scenarios for operating ML application using MLaaS and related functional requirements of MLaaS.

## II.1 Object recognition model developing with cloud computing environment

|  |  |
| --- | --- |
| Title | Objective recognition model developing with cloud computing environment. |
| Description | An AI software engineer would like to concentrate on developing and testing the machine learning models for object recognitions. However, the amount of labelled data set is not sufficient for developing and testing the model. In the cloud computing environment, the engineer can access the validated ML data set for object recognitions from the other engineers or companies. With the data given from the cloud, the engineer can build, train, and manage their own recognition model by testing the performance of the model.  In this use case, the engineer who is represented as CSN:MLMD can develop ML model in the cloud computing systems with validated data from the CSN:MLDP. The following steps shows the process of developing ML model.  1) CSN:MLMD registers the ML model to CSP:MLSP .  2) CSN:MLDP prepares ML data set with data labelling.  2-1) CSN:MLDP requests raw data to CSN:DP.  2-2) CSN:MLDP labelling the data with object detection algorithm.  NOTE 1 – The different detection algorithm can be adopted. The example of detection algorithm is rectangle detection and face detection with facial landmark.  3) CSP:MLSP requests ML data with the information of adopted detection algorithm  4) CSP:MLSP iteratively trains ML model with the ML data set until ML model is optimized  4-1) CSP:MLSP trains ML model with ML data set with default parameter.  4-2) CSP:MLSP evaluate trained ML model with validation data set.  4-3) CSP:MLSP change the model parameter and repeat 4-1) and 4-2) until model is optimized.  NOTE 2 – The ML model parameter can be different among ML models. The example of ML model parameter is Euclidean distance between the labelled data and validation data.  5) CSP:MLSP reports the performance evaluation to CSN:MLMD. |
| Role/Sub-role | CSN:MLMD  CSN:DP  CSN:MLDP  CSP:MLSP |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSN:MLDP searches data from CSN:DP and request data to CSN:DP. |
| Post-conditions  (optional) | CSP:MLSP stores the trained/optimized ML model for developing ML applications. |
| Derived requirements | – Clause 8.2 requirement (1,3,4)  – Clause 8.4 requirement (1,3,4)  – Clause 8.5 requirement (1,2) |

## II.2 Traffic speed prediction and monitoring service

|  |  |
| --- | --- |
| **Title** | **Traffic speed prediction and monitoring service** |
| Description | Traffic speed information on each road in a city are gathered from sensors. And then, traffic speed on each road after sometime, e.g. 15 minutes later, is predicted, and the prediction results can be used for solving traffic congestion in the city. The following is a procedure of this use case.   1. Raw data such as traffic speed on each road in a city are gathered by CSN:DP. 2. The gathered raw data are featured and pre-processed by CSP:MLSP, and the data are used for training and validating a model. 3. Some machine learning models developed by CSN:MLMD are provided, and those models are used to train models by CSP:MLSP. 4. Prediction model is trained and validated by using training data set. 5. Trained model is deployed and used for developing application. 6. Data for prediction are fed into the trained model or application, and prediction results are returned to the service user. The service user can use the prediction results. E.g., predicted traffic speed on each road are used to solve traffic congestion by controlling traffic signals. |
| Role/Sub-role | CSN:DP  CSN:MLMD  CSP:MLSP  CSU:MLSU |
| Figure (optional) |  |
| Pre-conditions  (optional) | CSN:MLSP can transform raw data used for predicting traffic speed. The transformed data are fed into trained prediction model.  The sensors for traffic observation are installed on each road or in each vehicle to gather speed and location data from the traffics passing the roads. |
| Derived requirements | – Clause 8.1 requirement (1,3,5)  – Clause 8.3 requirement (1,3,4)  – Clause 8.4 requirement (2,6)  – Clause 8.5 requirement (1,3,6,7,8,9,10,11)  – Clause 8.7 requirement (1,2,3,4) |

## II.3 Image recognition

|  |  |
| --- | --- |
| **Title** | **Image recognition** |
| Description | CSC:MLSU who is an application developer would like to provide image recognition in applications, e.g. animal recognition. In order to improve development efficiency, CSC:MLSU can just focus on UI interactions, and use MLaaS to implement image recognition function. The process would include the following steps:   1. The CSP:MLSP collect animal images with different formats from different the CSP:MLDP and storage those images. 2. The CSP:MLSP do some pre-process work to the images like: classify images by species (e.g. cats, dogs, pigs), converting images into the same format (e.g. convert jpg, jpeg, bmp into png), ranking images by resolution. 3. The CSP:MLSP send pre-processed images to the CSN:MLMD to develop ML model and using multiple worker node to get training accelerating. 4. After training, the CSN:MLMD registry image recognition model to the CSP:MLSP. 5. The CSP:MLSP perform image recognition for submitted image using the model registered by the CSN:MLMD. 6. The CSP:MLSP display the result of image recognition to the CSC:MLSU. 7. The CSC:MLSP provide accuracy, CPU usage and time cost of image recognition as a feedback to CSP:MLMD. |
| Role/Sub-role | CSC:MLSU  CSP:MLSP  CSN:MLMD  CSN:MLDP |
| Figure (optional) |  |
| Pre-conditions  (optional) |  |
| Post-conditions  (optional) |  |
| Derived requirements | – Clause 8.1 requirement (1,3,5,6)  – Clause 8.3 requirement (1,2,4)  – Clause 8.4 requirement (4,5)  – Clause 8.5 requirement (1,2)  – Clause 8.7 requirement (1) |

## II.4 Face recognition

|  |  |
| --- | --- |
| **Title** | **Face recognition** |
| Description | CSC:MLSU who is a door control system developer for a company needs to use face recognition for authentication for the employees and visitors. The most convenient way for the developer is to implement face recognition function by using the MLaaS service. The process would include the following steps:   1. The CSP:MLSU uses camera to collect videos taken from different angles of the employees. In order to realize liveness detection videos with spoofed faces on a screen are also collected. 2. The CSP:MLSU uploads those videos to the CSP:MLSP and splitting those videos into real face training set, fake face training set and testing set through the configuration reference point of the CSP:MLSP 3. The CSP:MLSP marks the position of face and eyes in the training sets. 4. The CSP:MLSP sends training sets to the CSN:MLMD and configurations values of hyperparameters to start developing ML model. During the developing, CSP:MLSP can get the learning status from the CSN:MLMD. 5. After learning process, the CSN:MLSP validates the trained ML models with testing set. If the validation results meet expectations, ML models will be registered and deployed. 6. When a visitor come to the company, CSP:MLSU captures video of the visitor. Then the CSP:MLSP perform face recognition for captured video using the model deployed. 7. The CSP:MLSP displays the result of face recognition to the CSC:MLSU. And the visitor is authenticated based on the result. |
| Role/Sub-role | CSC: MLSU  CSP: MLSP |
| Figure (optional) |  |
| Pre-conditions  (optional) |  |
| Post-conditions  (optional) |  |
| Derived requirements | – Clause 8.2 requirement (1)  – Clause 8.3 requirement (1,4)  – Clause 8.4 requirement (2,6)  – Clause 8.5 requirement (1,2,3,4,5,7,8,11)  – Clause 8.6 requirement (1)  – Clause 8.7 requirement (1) |

## II.5 Image Segmentation Model Development

|  |  |
| --- | --- |
| Title | Image Segmentation Model Development |
| Description | Image segmentation is a fundamental task for high level vision tasks such as auto-driving, robot navigation and etc. Recently developed deep neural networks can achieve performance with large amount of training data and powerful computation resources. Thus, it is a good choice for a CSN:MLMD to train and deploy the model in cloud computing environments. This use case includes the following key steps:   1. The CSN:MLMD designs the specific model params, such as network architecture and training loss function. 2. The CSP:MLSP defines the training data format and requests training data from CSN:MLDP. 3. The CSN:MLMD requests CSP:MLSP to prepare resources with necessary library installed (such as machine learning library) for model training. 4. The CSP:MLSP runs training process for the model committed by the CSN:MLMD and reports the training status including training and validation error. 5. After convergence, the trained model is registered and deployed in the cloud. |
| Role/Sub-role | CSN:MLMD  CSN:MLDP  CSP:MLSP |
| Figure (optional) |  |
| Pre-conditions  (optional) |  |
| Derived requirements | – Clause 8.1 requirement (1,3,4)  – Clause 8.5 requirement (1,2,3,4,6,9,11,12)  – Clause 8.6 requirement (1,3,4)  – Clause 8.7 requirement (1) |

## II.6 Generative Adversarial Model Development

|  |  |
| --- | --- |
| Title | Generative Adversarial Model Development |
| Description | Generative adversarial models can generate new samples whose appearance is consistent with those in the training dataset, e.g. generate an animal or a building. A simple generative model can be trained in an unsupervised manner, i.e. no labels are required. Neural network-based GAN (Generative Adversarial Network) models are one of effective methods for image generation. However, training a GAN is usually tedious and requires lots of tuning skills. It will be convenient if the cloud service provider can provide training and deployment service for a CSN:MLMD. Such a use case can be fulfilled with steps as below:   1. The CSN:MLMD requests specific training data from CSN:MLDP according to the task. For instance, if the model is to generate animals, then the training data should only contain large number of various animals. 2. The CSN:MLMD defines the network architectures of generative and discriminative modules respectively with corresponding loss functions. 3. The CSN:MLSP prepares computation resources and starts model training. Training process will stop if the convergence condition is satisfied or max iteration has been reached. Retraining can be executed if necessary. 4. After convergence, the model access method is returned to CSN:MLMD. |
| Role/Sub-role | CSN:MLDP  CSN:MLMD  CSP:MLSP |
| Figure (optional) |  |
| Pre-conditions  (optional) |  |
| Derived requirements | – Clause 8.1 requirement (1,3,4)  – Clause 8.5 requirement (1,4,5,6,7,9)  – Clause 8.6 requirement (5,6)  – Clause 8.7 requirement (1,5) |

Bibliography

[b-ITU-T H.752] Recommendation ITU-T H.752 (2015), Multimedia content provisioning interface for IPTV services.

[b-ITU-T X.1601] Recommendation ITU-T X.1601 (2015), Security framework for cloud computing.

[b-ITU-T X.1631] Recommendation ITU-T X.1631 (2015), Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services.

[b-ITU-T Y.2201] Recommendation ITU-T Y.2201 (2011), Requirements and capabilities for ITU-T NGN.

[b-ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), Security requirements for NGN release 1.

[b-ETSI GR ENI 004] ETSI GR ENI 004 V1.1.1 (2018), *Experiential Networked Intelligence (ENI); Terminology for Main Concepts in ENI*.

[b- *Jiawei H.]* Jiawei H., Micheline K., and Jian P. (2012)*, Data Mining: Concepts and Techniques Third Edition, MK, Morgan Kaufmann.*

[b- XinH]Xin He, Kaiyong Zhao, and Xiaowen Chu. (2019)*, AutoML: A Survey of the State-of-the-Art, arXiv preprint arXiv:1908.00709.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_