Rev. 2.1.0.0057-e25 (2022/02/04)

Machine Learning Quality Management Guideline

2nd Edition

February 4, 2022
(Japanese: June 30, 2021)

Technical Report DigiARC-TR-2022-01
Digital Architecture Research Center

Technical Report CPSEC-TR-2022002
Cyber Physical Security Research Center

Technical Report
Artificial Intelligence Research Center

National Institute of Advanced Industrial Science and Technology (AIST)

Foreword and Disclaimer

This English version of the Machine Learning Quality Management Guideline is a translation of「機械学習品質マネジメントガイドライン 第 2 版」(Machine Learning Quality Management Guideline 2st edition), published by AIST on July 5, 2021 in Japanese.

This guideline is assembled by the *Committee for Machine Learning Quality Management* in the National Institute of Advanced Industrial Science and Technology (AIST).

This guideline is non-binding in relation to laws and regulations/official guidelines. Provisions described as normative in this guideline have normative meaning only if the guideline is adopted voluntarily. This document is distributed on an as is basis, without warranties of conditions of any kind, either express or implied.

This guideline is developed under support from the New Energy and Industrial Technology Development Organization (NEDO).

Table of Contents

1. Summary of the Guideline

The content of this Chapter is informative. Contents included in this chapter that constitute the norms of the Guideline are described again in following chapters.

The overall structure of this summary is as follows. The chapters and sections in the list below show the location of the corresponding content in the main part.

- Section 1.1 explains the background and purpose of the Guideline.
- Section 1.2 presents expected ways of using the Guideline (Chapter 2).
- Section 1.3 analyzes the reasons why quality management of AI is difficult.
- Section 1.4 mentions an overall concept of quality management process, the concept which the Guideline is based on.
- Section 1.5 presents three viewpoints of *external quality* (quality viewpoints that do not depend on implementation methods and can be evaluated only through use), which the Guideline proposes to set as goals (Chapter 3).
- Section 1.6 complements the previous section and explains why some of components generally discussed as “qualities of AI” were not adopted in the previous section and views in the Guideline on how they should be addressed.
- Section 1.7 presents nine aspects of *internal quality* (quality aspects that depend on implementation method and can be managed by measurement or processes), which the Guideline proposes to consider as *means of the quality management* (Chapters 6 and 7).
- Section 1.8 presents an overall image of the development process on which the Guideline depends (Chapters 4 and 5).
- Section 1.9 clarifies the relationship between this Guideline and various external normative documents (Chapter 10).
- Section 1.10 explains the structure of the remaining parts of the Guideline.

1.1. Purpose and Background

The effectiveness of artificial intelligence (AI), especially machine learning technology, has been accepted in broad fields of applications such as manufacturing, automated driving, robots, health care, finance and retail business, and its social implementation seems to start to blossom. On the other hand, it is difficult to identify the cause when any accident occurs or to explain advantages of AI-based products to the amount of investments due to the lack of technologies to measure and demonstrate the quality of AI-based products or services. Consequently, wider acceptance of AI in the society is lagging, causing a big obstacle to the expansion of AI development business.

This document establishes a basis for quality goals for machine learning-based

products/services, and provides procedural guidance for realizing quality through development process management and system evaluations.

This document aims to enable providers of products and services to evaluate and improve the quality of their systems so as to reduce accidents and/or losses caused by AI malfunctions in the society. Furthermore, it enables stakeholders to express their product quality using provided norms, which can be used for both commercial purposes (e.g. quoting prices of their AI-based products) and social purposes (e.g. to express their responsibility to the society).

1.2. Expected target of the Guideline

The primarily expected users of the Guideline is providers of products and services which are constructed using machine learning (hereinafter referred to as *service developers*[1]) and system developers that actually implement products and services as software. For each product, the service provider and the system developer may be either a single entity that develops products or services and provides them to end users (referred to as *self-development entity*) or two separate entities depending on the sharing of responsibilities based on contract (sub-contracting or quasi-entrustment). Moreover, a service developer may sell implementation of machine learning components as a separate product. We primarily expect the Guideline to be used as a reference for these entities to share clear goals on required quality in accordance with situations in which products or services are used and to realize said quality throughout the system development process.

Furthermore, as a secondary usage, we expect the quality levels set by this Guideline can be referred to by end users for judging whether a particular system is safe to use. Also, the Guideline is expected to serve as a technical starting point for specifications and evaluation/certification the quality.

The Guideline is described in a generic way that it can be applied to as broad use cases of machine learning technology as possible. For each of specific application area, one can pick required parts of this documentation to make a specialized guideline. In addition, we expect each developer may choose to make their own specific version of development guideline to use, based on the document. The research project currently plans to publish some examples of such document as reference. The following list shows some possible cases of use of the Guideline as an example.

- When an organization for AI-based system development is established (contract, etc.)
  - An entity requested to develop a machine learning-based system or a machine learning component which constitutes that system (referred to as *development*

1 In cases where software developers implement systems in advance and sell these as packaged or customized products, such developers are treated as service providers in this guideline.

entruster[2]) concretizes the Guideline in line with an application and designates it for an entity to which development is entrusted (referred to as *development entrustee*) as specifications that serve as contract requirements.

- An entity that is to be a development entrustee refers to the Guideline as a standard for process management to guarantee the quality for a development entruster and use them as a ground or reference to calculate man-hours and the price for order.

– Design and development
  - An entity that designs and develops a machine-learning based system or a machine-learning component (development entruster or development entrustee) uses the Guideline as a standard for process design, system design and quality management.

– Society use
  - A self-development entity or development entruster bases its self-compatibility declaration on the Guideline with respect to quality requirement as a social norm when a system is provided to the society or used for its own business.
  - The Guideline serves as standards for social consensus on the quality of machine-learning based systems in the future.
  - The Guideline serves as standards for the design and operation of a third-party assessment system on the quality of machine-learning based system.

This Guideline is currently applicable for machine learning based systems that implemented with supervised learning. Although the basic concept can apply to other implementation methods such as unsupervised learning, semi-supervised learning and reinforcement learning, a specific way of treating them will be added in future revisions.

1.3. Challenges behind machine learning quality management

The implementation of artificial intelligence by machine learning is, if simply said, a sort of software. However, a conventional concept of software quality management is not technically feasible to improve and maintain the quality of machine-learning based systems. This section sorts out challenges related to differences between AIs and conventional software from several viewpoints.

2 The “Guideline for Contract on Use of AI and Data” compiled by the Ministry of Economy, Trade and Industry of Japan defines *development entruster* in this Guideline as “user”, and *development entrustee* as “vendor” or “SIer” (system integrator), as it is written mainly for B2B (business to business) development agreements. This guideline, however, uses the term *user* only for end users of products and services, who are finally affected by the quality of the products.

1.3.1. Importance of environmental analysis

Machine-learning based products and services are often used under environmental conditions whose complexity is difficult to be understood by humans. Compared to regular programs for which humans have to analyze and identify complexity of all environmental conditions and implement every program codes as rules for judgment, there are high expectations on machine learning technology capable of automatically establishing rules for judgment based on data without having to identify detailed environmental conditions through analytical works by humans as an efficient means for implementing systems that operate under highly-complex environmental conditions.

On the other hand, from the viewpoint of ideal quality management, it is desirable to analyze environmental conditions as precisely as possible and understand risks at the early stage of system design especially from the viewpoint of quality management of systems whose compatibility to rare environmental conditions is called into question. If a machine-learning technology is used for these systems, analyses on environmental conditions that used to be carried out at the time of implementing program codes in the past would be omitted. This is the reason why environment analyses at the early stage are important.

Therefore, it is important to consider how deeply the status of use and environmental conditions are analyzed in order to balance between two characteristics (efficiency of implementation and environmental compatibility) at an early stage of system design including differences from the development of conventional software.

1.3.2. Lifetime-long requirement for risk assessment

Generally speaking, a system that operates in real world and constitutes so-called cyber physical systems, hereinafter referred to as *CPS*, has risks originating from external environmental changes in addition to risks that exist in regular software systems. Different from the implementation of regular software capable of theoretically analyzing and simulating unfamiliar situations to a certain degree and taking countermeasures, a machine-learning based system established based on real data may not be capable of dealing with significant changes in data trends originating from changes in our surroundings (although we have some anticipations on its generalization capability).

Moreover, a machine-learning based system may be designed in a way that it can attain practical quality only after additional learning using real data at the operation stage.

From this viewpoint, it is often necessary to introduce a continuous process lifecycle which integrates development and operation (DevOps) so that it can respond to situational changes after the system's initial operation. We need to plan a lifecycle that includes operation-time updates from the early stage of the development planning and to reflect its requirements into both the operation plan and the contracts between stakeholders.

However, ensuring a certain level of quality at the starting time of operation is often necessary, even if quality management will be carried out continuously. It is true particularly for systems with risks that compromise safety. From this viewpoint, the Guideline focuses on quality tests conducted in stages from the implementation to the beginning of operation. Moreover, effects of risks and the required quality level could be changed by varying the form of system operation at the early and full-blown stages (for example, different levels of possibility of risk avoidance due to presence or lack of monitoring or intervention of the system by humans). In this case, it is required to synchronize the time when the form of operation is changed and the time when quality management goals are changed.

Furthermore, if the system implementation is updated at the time of operation, there is a risk that sometimes the quality deteriorates (called regression in software engineering) even if that update aims to improve the quality. Therefore, we need a certain form of quality monitoring and countermeasures at the time of operation, but they vary depending on forms of development and operation. Section 4.3 sorts out some possibility for such operation models.

1.3.3. Quality assurance depending on data

A request for *quality of data* used for building machine learning is often mentioned in data-oriented development. The Guideline assumes that the quality of data itself is not the ultimate goal of quality management but rather the cause of quality deterioration or means for ensuring quality. Of course, it is almost indispensable to ensure a certain level of data quality to actually ensure the machine learning quality through the lifecycle process management such as the Guideline. In addition, when a machine-learning based system is developed by sharing works, learning data may be sold/bought or shared for development. In such a case, there is a room for discussion by treating *data quality* as an independent character. Moreover, there has been pointed out at recent academic meetings that there are security risks such as contamination of learning results due to intentional injection of improper data. Effects of such improper data cannot be detected by simple numerical evaluations but require checks on data sources from broader perspectives.

The Guideline is supportive of ensuring the quality by means of numerical evaluations to the extent as possible, but (at least now) we need to secure the quality through qualitative data quality management as well.

1.4. Basic concept of quality management

(note) The Guideline defines *quality in use* as quality to be provided to end users in the whole system. On the other hand, *external quality* and *internal quality* are examined at component level[3] which will be combined into a single system.

External quality of each component is quality required as a part of system from an objective perspective. It includes, for example, security, reliability, and consistency. This external quality includes both qualitative and quantitative qualities and cannot always be expressed in single measurable indicator.

On the other hand, *internal quality* of each component refers to quality as a unique characteristic which is specifically measured or evaluated through acts of development including design when the component is created.

Based on this concept, Figure 1 explains hierarchical quality model in which external quality of each component is realized through its improved internal quality and is required for achieving internal quality of the element in one layer outside. *External quality* of the whole system is realized and provided by a provider of products/services for the sake of *quality in use* viewed from their end users.

[3] The term *internal quality* roughly coincides with the "internal measure of the quality" in ISO 25000 series. The concept behind the *external quality* is not directly associated with "external measure of the quality", because the guideline assumes that external quality is not generally satisfactorily measurable as a numeric quality indicator. Instead, this document ensures satisfaction of an external quality *level* through assessments of internal qualities.

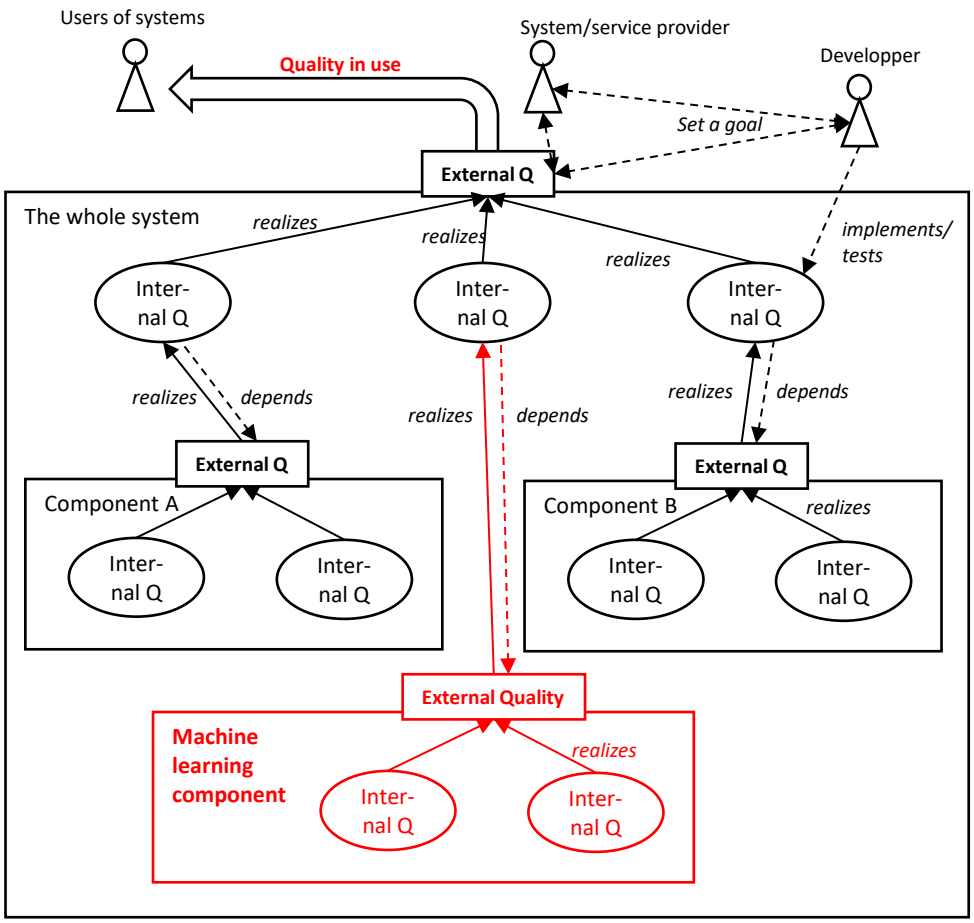

Figure 1: Hierarchical quality model

In the Guideline, *quality* of a machine-learning based system itself is understood by 1) *quality in use* expected to be satisfied in the whole system in use, 2) *external quality* expected to be satisfied in components of the system built by means of machine learning, and 3) *internal quality* which constituent components built by means of machine learning uniquely have. *Quality* is understood as what satisfies a required level of *external quality* through improving *internal quality* of machine-learning components and realizes *quality in use* of a final system (Figure 2).

Three viewpoints listed in Section 1.5 were set as external qualities of machine-learning components (target of quality management). We focused viewpoints unique to machine-learning components as internal qualities to achieve said external qualities and extracted eight viewpoints listed in Section 1.7 at this stage. The quality goals were categorized by level in relation to the three viewpoints for external qualities, and the performance goals were set for each of the nine viewpoints for internal qualities in accordance with said categorized levels in order to achieve those goals through various technologies and development process management. This is a basic concept of quality management in the Guideline.

Since qualities in use of the whole system to be realized ultimately have different focuses

depending on its application, it is not specified specifically in the Guideline (See the following supplement).

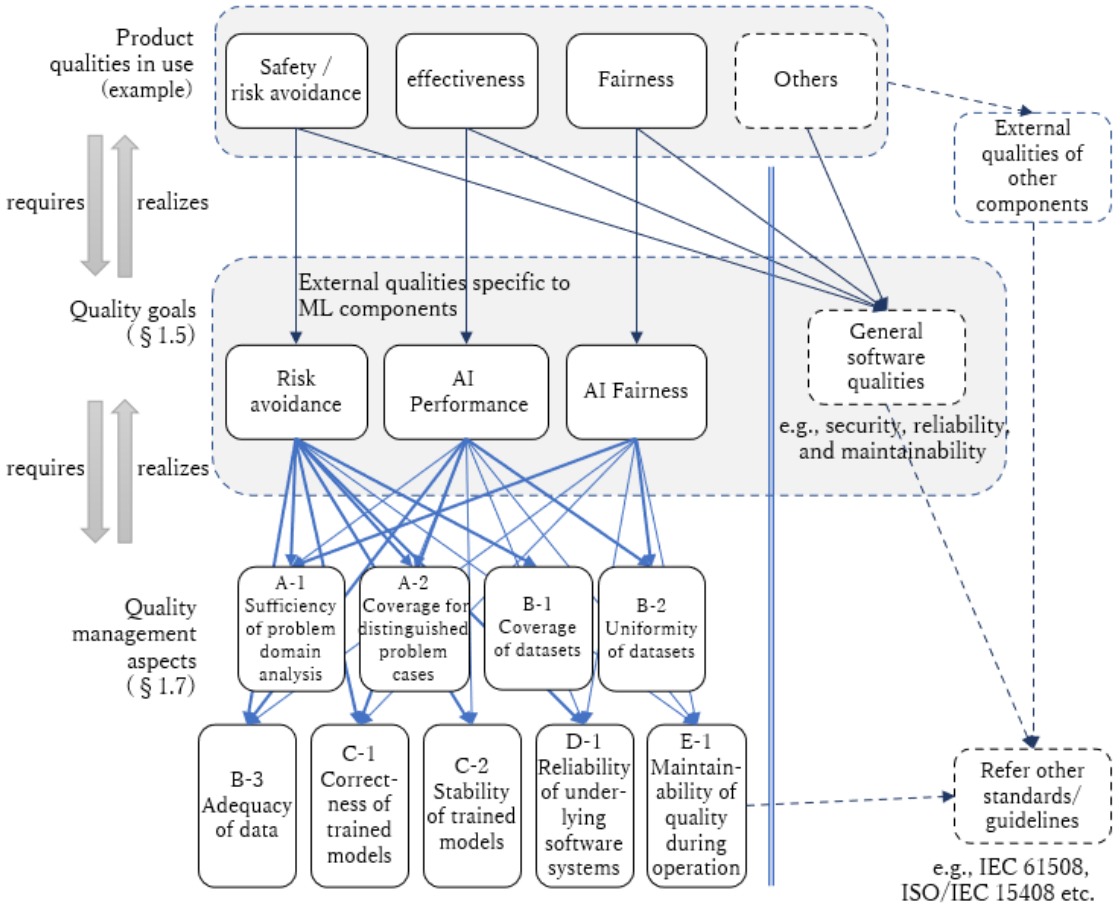

Figure 2: Overall structure of realization of product quality

Example 1) A module to recognize objects based on front images installed on a self-driving car is envisioned. One of qualities in use of automobiles is *safety for avoiding collisions with obstacles under all environmental conditions under which automobiles can be driven*. In order to realize this, the object recognition module must have a characteristic that *it can correctly recognize objects in all possible climate conditions, times etc.* as an external quality. Internal qualities required to realize this include completeness of learning conditions. This is realized by the process of, e.g., preparing machine learning training data.

Example 2) AI to estimate stock prices included in automated stock trading services is envisioned. One of qualities in use of the whole service is *maximization of profits*. A module to estimate stock prices must have a characteristic that *minimizes errors in stock price predictions and maximizes the total of envisioned trading results* as an external quality. Internal qualities

required to realize this include, for example, the precision of reasoning of machine learning This is realized by optimizing machine learning training parameters.

(Supplement) When a product or service is developed using strict processes to evaluate safety and reliability of the whole industrial systems such as IEC 61508 [12] or IEC 62278 [15], a requirement for external qualities is of course identified in relation to machine learning components in the design process of whole conventional systems.

On the other hand, if products or services are used for IT services that have strong request for machine learning technologies, an expected level of risk is not as high as that of industrial systems in many cases. Therefore, it may not be necessary to apply a strict risk management process to the whole system development.

Moreover, if we expect machine-learning components to "make a wise decision" in any way as an overall use of AI technology, the three viewpoints listed in this Section may often require somewhat direct response to external qualities and qualities in use of the whole system. The above two cases illustrate specific cases.

Based on such observations, Figure 2 is described in a way that qualities in use of the whole system and external qualities of machine-learning components are paralleled. Moreover, three qualities in use (safety, effectiveness, and fairness) are exemplified.

1.5. External quality characteristics to be achieved

In conventional systems, quality requirements on most software components are *correctness in terms of given design specifications*, whatever the quality-in-use required to the whole system is. It is because most of the quality-in-use requirements are considered at the system design stage, not at the implementation stage.

On the contrary, in many cases, implementors of machine learning components are required to consider some quality aspects closely related to the quality-in-use.

This document identifies the following three properties as the quality goals specific to machine learning components.

1.5.1. Safety / Risk Avoidance

The *safety*, on the machine learning component level, is a property to reduce possibilities of generating undesirable, probably harmful outputs from a machine learning component. The following are some specific examples related to the safety property:

- Oversight in object recognition for automated driving and false recognition of type thereof;
- Oversight of foreign objects in the detection of contaminants in a food factory's

production line; and
– An order exceeding the acceptable level in automated trading of securities due to illegal input (spoofing)

We have set seven levels of safety, varying from the one that affects many human lives and continuity of businesses to the one that only causes minor losses of opportunities for profit (Section 3.1). Since the characteristics of safety are closely related to the properties handled in safety specifications for conventional systems, we have set four levels (Levels 4–1) corresponding to the existing specifications (e.g. SIL of IEC 61508-1 [12]) in response to strong industrial requests taking the combination and affinity with those specifications into consideration. On the other hand, typically for the application of machine learning to IT services and smart devices, we added three levels from the practical point of view, because only minor damages that are not considered as risks in conventional industrial products are envisioned in many cases so that these requests for quality are categorized into the same level (Not Applicable) in the existing safety specifications.

Generally, in machine learning systems, it is not practical to strictly guarantee a characteristic that *they always operate safety at all times* and machine-learning components themselves may not achieve qualities in use required for the whole systems. When a system is actually built, its realization depends in many cases on *safety assurance valves* by implementing peripheral software not on machine-learning components. The Guideline recognize requests for qualities in use of the whole systems and requests for external qualities of machine-learning components separately in the same way as conventional specifications for safety and set levels required for external qualities of machine-learning components based on risk assessments on the whole system and its configuration (Section 5.1.1.5).

1.5.2. AI performance (usefulness)

As the second axis of characteristics, we focus on applications to fields in which the usefulness of machine learning functions is emphasized and categorize it as the axis of characteristics of *AI performance*. A typical application in which priority is given to AI performance rather than to risk avoidance includes a scenario in which higher average performance is required than negative effects caused by individual misjudgments such as request forecasting by retail stores and forecasts of investment decisions.

In some cases, both *AI performance* and *safety* are required. For example, on a price forecasting application, an excessive purchase could cause more indirect negative effects on management environment than expected, resulting in big economic losses that cannot be assessed based only on the average value of profits. In this case, it is necessary to clarify an optimal middle ground of two axes of characteristic, *AI performance* and *risk avoidance*.

Since specific targets to be achieved of AI performance (Key Performance Indicator (KPI)) differ from one application to another, three levels were set from the viewpoint of how much

the achievement of those goals is required.

1.5.3. Fairness

A kind of social norms or ethics is often required for AI. From the engineering (not humanities) viewpoint, many of these social norms boil down to various existing characteristics of qualities in use and the process of their setting. Under these circumstances, we can extract *fairness* as a quality viewpoint specific to AI, which has not been taken into consideration in the past, by using a recent statistical technique.

Although conventional software has to make a *fair judgment* in many cases, its realization completes mainly when examinations are made in the design stage. Therefore, other quality characteristics such as *reliability* (software operates correctly as designed) were emphasized in the implementation stage. However, in machine learning, probabilistic and statistical behaviors are included in the implementation process and learning results so that a prior examination is not enough to ensure fairness, and machine-learning components (software elements) have to directly realize *fairness* as a quality.

From this perspective, the Guideline picks up *fairness* as the third external quality characteristic. In this guideline, *fairness* is defined as the absence of different treatment of inputs according to factors other than conditions that the machine learning element should use as a basis for judgment, with respect to functional requirements specified at the level of the system or machine learning element. Examples of such fairness would be cases in which it is explicitly or implicitly required that there is no bias in the treatment of racial differences or gender in, for example, health insurance estimates or personnel hiring scoring. More precisely,

- *Fairness* is defined as the fact that multiple possible inputs or people who produce inputs are not *treated differently, due to differences in situations other than those defined in requirements*, under some defined criteria, compared to the requirements for situations that should be used as a basis for judgment, which are defined as functions required for judgment, classification, etc. performed by a system or component.
  - Depending on the functional definition of the system, *situations other than those defined in requirements* may be specified in a reverse way, as *differences in specific situations that should be treated equally*. The *situation* can be an attribute that is explicitly included in the value of the input to the machine learning element, or it can be a hidden feature that is not included.
  - The criteria for *different treatment* can be variously specified depending on the strength of the fairness requirements of the system, for example, *no intentionally different treatment, no statistical difference*, or *algorithmically guaranteed same treatment*.
  - How to elaborate this definition of *fairness* for individual cases is discussed in

more detail in Section 8.5 of this document.

– In contrast, the term *bias* refers to the statistical correlation of the output of an implemented machine learning element with various characteristics of the input. By this definition, unless the output is a constant or random function, it will have some bias, but some of the unintentional biases may cause damage to fairness.

Four elements (FAST (fairness, accountability, sustainability, transparency)) are generally pointed out as social requests for machine-learning based systems. The Guideline focus firstly on fairness which can be directly analyzed as a statistical characteristic.

More detailed discussions on Fairness property, including its background, are given in Chapter 8.

This guideline does not consider the property that certain levels of risk avoidance and AI performance are required for all attributes as fairness property, but instead as a property of risk avoidance and AI performance. For example, a request for quality concerning the effect of differences in gender or physical constitutions in protective devices for safety is sorted out as a *request for risk avoidance so that they are safe for all genders and physical constitutions*, not as a *request for no differences in safety due to gender difference*. This is because a technical way of realizing fairness can be sufficiently balanced out with other performance indicators and it is not desirable to solve the problem by improving fairness for some attribute values by reducing safety for the other values.

Furthermore, as regards this external quality characteristic, we examined *which aspects of fairness are measurable and can be subject to quality management* and *how can those aspects be realized in machine-learning based components* in the Guideline. *What kind of fairness realizes social justice in a specific system* should be examined in advance outside the scope of the Guideline. These social norms and ethics will be discussed further in Section 1.6.3.

1.6. Possible other aspects for AI quality

The Guideline set forth that the quality is managed with a focus on three viewpoints (risk avoidance, AI performance and fairness) as described above. Generally, various viewpoints are under discussion concerning the “AI quality”. In this section, we summarize our thoughts on the relationship with this guideline for some perspectives other than the three we have organized so far.

1.6.1. Explainability of AI

The *explainability* of AI systems is sometimes emphasized as one of the components of trustworthiness. In fact, there are several aspects to considering explainability as a quality. First of all, from the perspective of *explainability of being able to use the product with confidence*,

or in other words, explainability of quality, this guideline considers this as a property required for quality management rather than quality itself. Throughout this guideline, quality management is viewed as a series of activities that enable quality management activities to be explained and accepted afterwards with conviction, and explainability from this perspective is the very purpose of this guideline.

Next, from the standpoint of the explainability of the behavior, explainability in this sense can be considered as one of the means to achieve quality explainability, which is the objective of this guideline. Indeed, if the behavior of a machine learning system is fully explained, it is highly likely that its behavior can be logically analyzed and its quality explained in the same way as an ordinary program. And even if the behavior cannot be perfectly explained, machine learning elements with a simple structure may be expected to be easier to explain the content of the behavior, and at the same time have lower quality concerns that may lead to different behavior than expected. On the other hand, however, when the environment is complex as described in section 1.3.1, even ordinary software programs that would logically be perfectly explainable often do not necessarily lead to reliability, and the explainability of behavior and the explainability of quality are not always considered to be the same thing.

1.6.1.1. Explainable AI

The idea of *explainable AI* is often referred to as a form of achieving explanatory behavior. Once the technology for explainable AI is well established, where the behavior of AI can be logically re-described and organized at a level that is sufficiently understandable, it is expected that the achievement of various qualities can be logically explained from the standpoint of quality management. However, at least at present, such technology is still under research and has not reached the stage where it can be adopted stably and universally as an all-purpose means to achieve quality. Of course, if the technology can be applied, it can be a powerful means of achieving quality in relation to these guidelines. In particular, it could be a strong means to obtain explanations of internal quality specific to fairness (Chapter 8) and stability of machine learning models (Section 6.7, Internal Quality C-2). It should also be noted, as mentioned above, that even if the algorithmic computational processing of machine learning elements can be perfectly explained, it does not always explain their quality in the real world.

1.6.1.2. Transparency

For the same reason, we consider transparency transparency to be one of the means of achievement for quality management. At least in the field of machine learning, transparency of behavior is a rather similar property to explainability of behavior, and may be useful in the analysis of fairness and stability. Also, in applications where the resulting machine learning

model is not used as it is, but is rather designed logically based on the parameters inside the obtained model and implemented as a rule base or as an ordinary program (not considered as a machine learning model in this guideline), transparency and explainability as an ordinary program may be considered.

1.6.2. Security and privacy

The security of machine learning based systems is discussed from the following two perspectives: (i) the attack surfaces, threats, and vulnerabilities that have been addressed for conventional information systems, and (ii) the attack surfaces, threats, and vulnerabilities specific to machine learning.

The security in terms of conventional information systems consists of the three principles: *confidentiality*, *integrity*, and *availability*. These should be managed based on conventional standards, e.g., the evaluation assurance level (EAL) of ISO/IEC 15408 [3]. As for the confidentiality of data, the stakeholder may need to take the measures for privacy protection that have been designed for conventional information systems, such as the anonymization of personal data under the legal regulations (e.g., GDPR).

On the other hand, the security against the attack surfaces, threats, and vulnerabilities specific to machine learning mainly involves confidentiality and integrity.

- A *poisoning attack* is an attack to degrade the integrity of a machine learning based system by manipulating the trained machine learning models or the training data during or before the training phase.

- An *evasion attack* is an attack to degrade the integrity of a machine learning based system by inputting adversarial examples to the system during the operation phase. The evaluation and countermeasures for evasion attacks involve the *stability* of machine learning models (Section 1.7.7).

- A *data privacy attack* is an attack to degrade the confidentiality of training data. In this kind of attack, an adversary may be able to obtain information about training data by interacting with the trained machine learning models or the system. This may result in the leakage of personal information and privacy.

It should be noted that security countermeasures against these attacks may degrade the accuracy of the machine learning components and thus the AI performance of the system.

The developer needs to perform security risk assessments and take countermeasures on a regular basis. Then the scale and the priority of each countermeasure should be determined by the risk assessment.

In Chapter 9, we summarize the main issues in the security of machine learning based systems and machine learning components.

1.6.3. Social aspects such as ethicalness

As qualities required for machine-learning based systems, ethicalness and social validity of results and judgments made by machine-learning components may be included. For example, in fields closely related to personal information (prior scorings for hiring, crime forecast, etc.), it may be required to guarantee that differences in sex or races do not affect judgments legally or socially. Moreover, in fields that are likely to cause human and economic damages, there may be a case where the validity of possible judgments with different risks is called into question (a judgment made by a self-driving system under a situation where some human damage is inevitable).

The Guideline do not directly examine what kind of judgment machine-learning components should make to have social validity in those cases, since it should be sorted out by humans in advance as a part of required definitions at the beginning of system development. Then, a machine-learning based system draws out processing results with a high probability to the extent possible toward “correct” output sorted out by humans, and this is considered as a request for qualities in use.

As such, *fairness*, which has not been treated as a direct external quality in conventional software engineering in particular, is listed in Section 1.5.3 and Chapter 8 as one of the external quality axes. As for ethical aspects such as fairness and transparency, we summarize some international efforts in Section 10.2 for reference.

1.6.4. Limit of response to complex external environments

As a general problem of AI, discussions on the limit of complexity of environments where AI can be utilized draw attention. Machine-learning based systems are often incorporated into a part of so-called cyber-physical systems in open environments such as streets and public spaces. Therefore, it is impossible for AI to make expected judgments under all environmental conditions, if ultimate conditions are included. This issue is inherently common not only to machine learning but also to devices and software that operate in open environments. Conventional specifications for reliability engineering such as IEC 62998 [16] seem to intend to tackle this issue somehow.

The Guideline follow the overall concept of past specifications. The adequacy of risk analysis and system design required for realizing the quality in complex external environments becomes subject to examination of quality management methods with regard only to differences in characteristics from conventional software and points to remember for unique analysis and design originating from those differences.

1.7. Internal quality characteristics subject to quality management

In the Guideline, the following nine characteristics, in five categories, were extracted as characteristic axes of quality management, for the achievement of three external qualities described above. Section 11.1 describes the outline of analyses leading to this extraction.

- A: Designing quality structures and data sets
  - A-1: Sufficiency of problem domain analysis
  - A-2: Coverage for distinguished problem cases
- B: Data set quality
  - B-1: Coverage of datasets
  - B-2: Uniformity of datasets
  - B-3: Adequacy of data
- C: Quality of machine learning models
  - C-1: Correctness of trained models
  - C-2: Stability of trained models
- D: Quality of software implementation
  - D-1: Reliability of underlying software systems
- E: Operational quality
  - E-1: Maintainability of quality in operation

This section mainly explains internal quality aspects for the two external qualities of *safety* and *AI performance*. Quality aspects for *fairness* will be summarized in Chapter 8, in addition to this section (and Chapter 6), as there are aspects that require specific consideration for fairness.

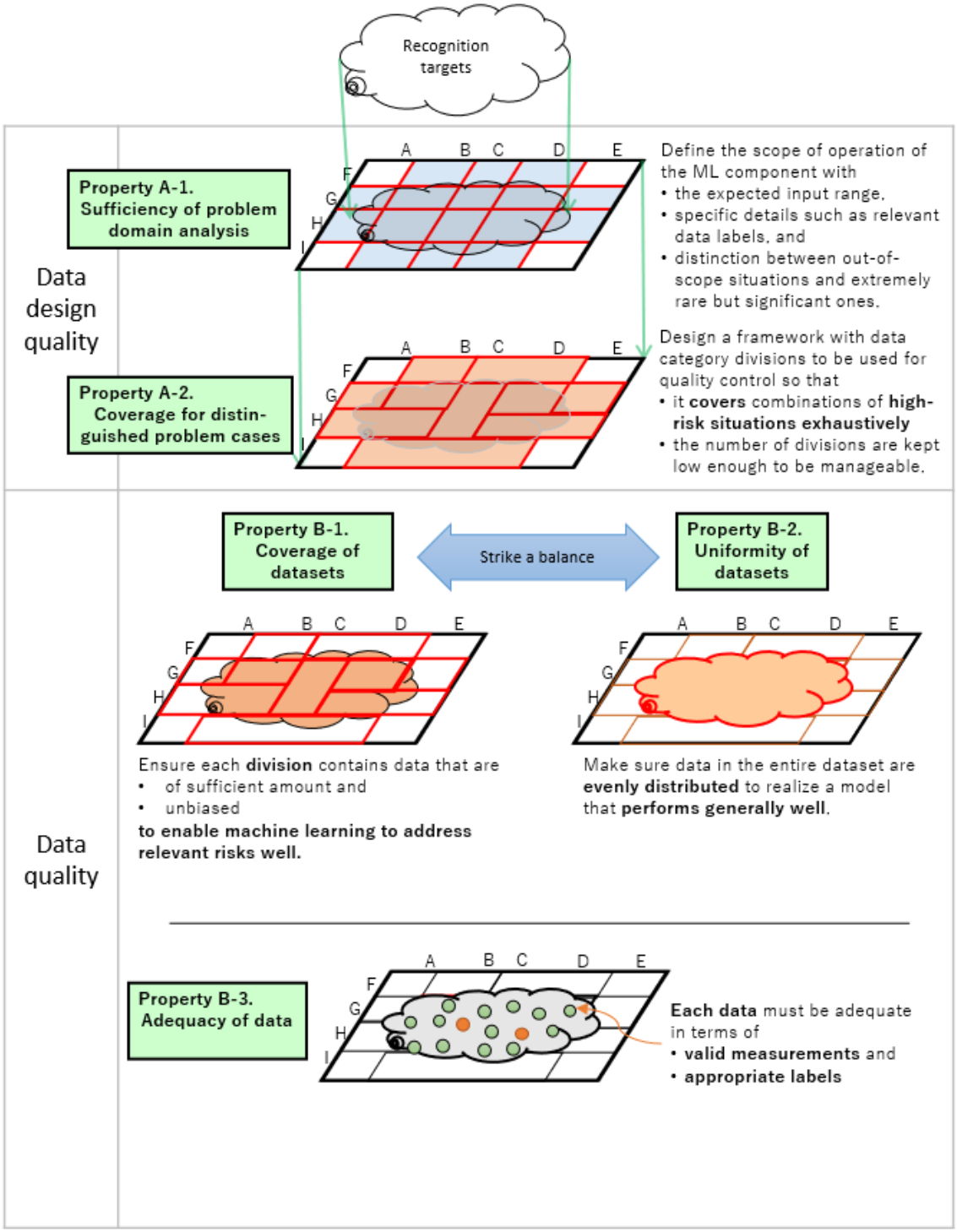

Figure 3: Internal quality characteristics focused (1)

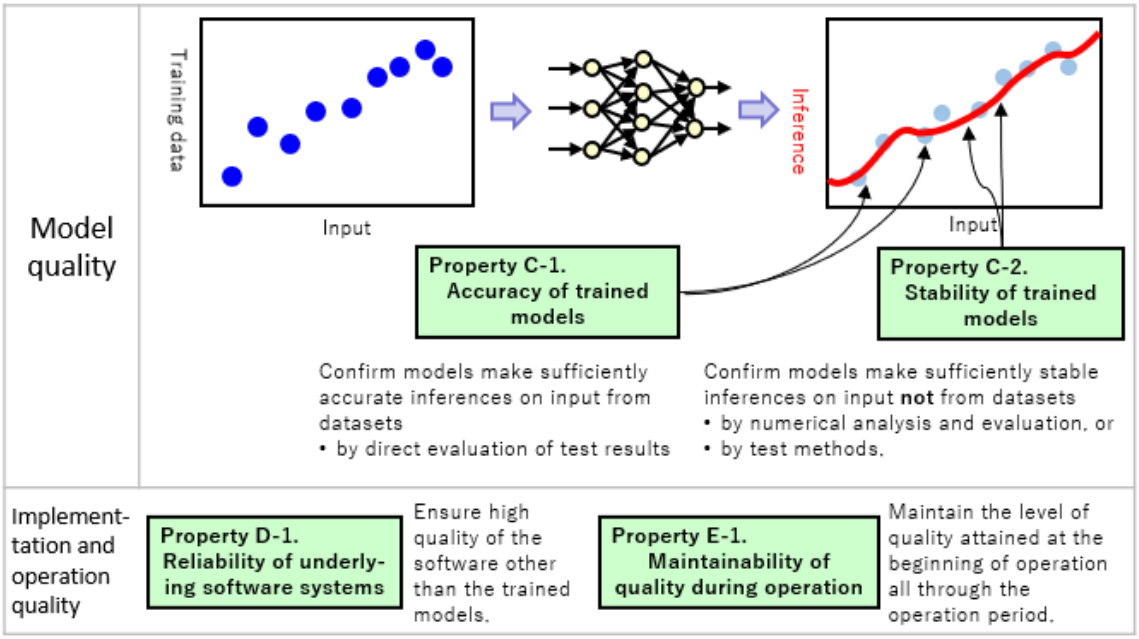

Figure 4: Internal quality characteristics focused (2)

1.7.1. A-1: Sufficiency of problem domain analysis

First, we define *sufficiency of problem domain analysis* as the fact that the analysis of the nature of the actual data, expected to be input to the machine learning element during operation, corresponds to the real-world usage of the machine learning application system, and that the results of the analysis cover all expected usage situations.

There are different ways of setting a specific axis for data analysis. The basic concept of the Guideline envisions a concept of feature tree in conventional software product line engineering and a more simplified method of classifying and organizing *the axis* as itemized independent conditions and capturing specific uses as their combination (See the following examples). Moreover, we conduct at this stage both top-down analysis from the request side such as risk analysis/failure mode analysis and bottom-up analysis consisting of preliminary data analysis in the PoC (Proof of Concept) stage to sufficiently sort out possible situations in which external qualities change (deteriorate).

(Example 1)

We can write down the following situations which a self-driving vehicle running outdoor might encounter in machine-learning components that recognize traffic lights based on images.

(Please be reminded that this example is not sufficiently comprehensive for actual application)

Significance of display: Green, yellow, red

Time zones: Day, night
Weather conditions: Sunny, cloudy, light rain, heavy rain, snow, fog, etc.
Others:

For example, the situation in which "the traffic light is red on a clear night" is one *combination of situations*.

(Example 2)
We can write down how a machine-learning component that predicts the sales trend of a retail store is used as follows (this example is not comprehensive, either).

Day: Monday, weekday, Friday, Saturday, Holidays
Weather conditions: Sunny, cloudy, light rain, heavy rain, snow
Time zones: Morning, before noon, noon, afternoon, evening, night, midnight
Season: Spring, summer, fall, winter
Neighboring events: Yes/No

In this example, one combination could be "there is a neighboring event on a holiday morning under winter clear sky".

The sufficiency of this requirement analysis deals with analysis of risk factors in conventional software and test designs to include those risk factors when a black-box test is conducted. It is also an important characteristic that establishes a unit to grasp and check the quality. On the other hand, when machine learning is implemented, minor characteristics above a certain level may be left to learning processing in the training stage, and a person who implements the system may not give detailed instructions on how to judge individual conditions. We can say that these are the biggest benefits. Moreover, in some cases, the implementation by machine learning shows better performance than that by humans. From this viewpoint, the configuration of *details to be included in requirement analysis* becomes a very important point in examining an implementation strategy of machine-learning components taking the quality into consideration.

Furthermore, when this type of analysis is made, we may recognize a situation which we do not encounter in real life and an extremely rare situation in which it is not practical (it is not required to respond) to guarantee operation (e.g. snowfall in summer) or a rare situation that does not appear in training data to be collected but it is required to respond to (e.g. snowfall in spring in the Tokyo area). Specifically, it is very important to distinguish those two situations in any system for which risk avoidance is required and the distinction is directly connected to design of safety/robustness of the whole system. At this stage of considering the *sufficiency of problem domain analysis*, it is also important to identify situations that cannot

happen in real life and eliminate them from examinations in later stages, in the process of identifying rare situations (cases) difficult to be found based only on data collected from real life that require response and designing corresponding systems.

A specific concept of configuring these situations will be explained in Section 6.1 in more detail.

1.7.2. A-2: Coverage for distinguished problem cases

Assuming sufficiency of problem domain analysis, sufficient examinations of data design in response to various situations to which systems have to respond, for collecting and organizing sufficient training data and test data, are required as *Coverage for distinguished problem cases*.

In an extremely simple system, it is enough to mention that corresponding data to all *combinations of situations* identified in the problem domain analysis described in the previous section included in training datasets and test datasets. However, if a situation in which a system is envisioned to be used is complex, the number of possible combinations becomes enormous. Therefore, it is not practical to cover all combinations with datasets (for example, only in the simple case mentioned in the above Example 2, the total number of combinations is 700. In a real case, this number is envisioned to reach 10,000). In this case, it is required to check completeness with a rough granularity level that covers several situations to sufficiently deal with combinations of detailed situations in which any danger or reduction in performance is likely to occur. There is a concept of *coverage criteria* applied to design of tests in the field of software engineering which aims to achieve practical and sufficient operational completeness by selecting appropriate means for each application.

1.7.3. B-1: Coverage of datasets

Next, we require that a sufficient amount of data (especially, test data) is given to each *combination of situations to be supported* designed in the previous section without any missing situation, to be called as *coverage of datasets*. Although the property mainly concerns datasets for testing and validation for the purpose of quality management, to achieve targeted levels of qualities, it matters also on datasets for training.

When regular software is developed, the details of all features in real world which software operation depends on is grasped in any point from problem domain analysis to implementation and reflected ultimately as conditional branching or calculation formula in programs. However, when machine-learning components are built, more minute situations than a certain degree are not grasped explicitly as *feature quantity* or *ground truth labels* of training datasets and are only included implicitly in training datasets. They will be reflected in final operations throughout the training stage of machine learning. The purpose of

configurating this axis of characteristic is to guarantee that the shortage of learning due to the shortage of data or any oversight of learning in specific conditions due to biased data does not occur in any situation or case identified in requirement analysis or data design.

(Example 1)

In the case of recognition of images of traffic lights mentioned earlier, this characteristic means that all forms of traffic lights in each prefecture, their height and distance between them, and road conditions around them are included as data without any bias and that training is not carried out based only on limited data of a specific city.

(Example 2)

When image recognition AI that aims to recognize *cats* from small animals around town is to be built and individual characteristics of cats such as breed and size are excluded from recognition, this characteristic means that sufficiently diverse images of *cats* and *other small animals such as dogs* are made available as data and that training does not only target specific breeds (e.g. calico cats) in environments where the product is expected to be used.

1.7.4. B-2: Uniformity of datasets

A concept contrary to *coverage* mentioned earlier is *uniformity* of data in relation to the overall assumed input data. When each situation or case in datasets is extracted in accordance with the frequency of its occurrence in whole data to be input, data is considered as *uniform* (Figure 5). Balance between this uniformity and coverage sometimes needs proper attention and evaluation. The prediction accuracy of machine-learning technologies is generally supposed to improve by using samples extracted uniformly in relation to input environments as training datasets. However, the coverage of situations explained in the previous property may be emphasized depending on actual application and quality characteristics required therefor. It is necessary to consider priority or compromise between coverage and overall uniformity carefully.

When risk avoidance is strongly requested, sufficient training data is required for high-risk situations that have to be avoided by making correct judgments. If such a rare case occurs and you intend to train data by ensuring a sufficient amount of data in relation to that rare case and maintaining the uniformity to all other situations, the necessary amount of data might be enormous. In this case, priority may be given to training of rare, high-risk situations.

On the other hand, when overall performance (AI performance) is requested, certainty of inference results in other cases might deteriorate by giving more priority to the training of rare cases than the actual probability of occurrence, thereby resulting in the deteriorated average performance as a whole. In this case, standards for coverage for each situation mentioned in the previous section are not appropriate.

Moreover, when fairness is strongly requested, *what kind of fairness* is requested may change our decision. For example, you may choose to give the cases artificially-equal training by artificially processing data (selection or addition of data) or let the cases learn at random in line with the distribution of extracted training data.

Since the two viewpoints mentioned in Section 1.7.3 and this Section may be compatible or incompatible, you may need to adjust appropriate training data to strike a balance at an adequate level. Moreover, different characteristics may be sought in the training stage and the quality test stage.

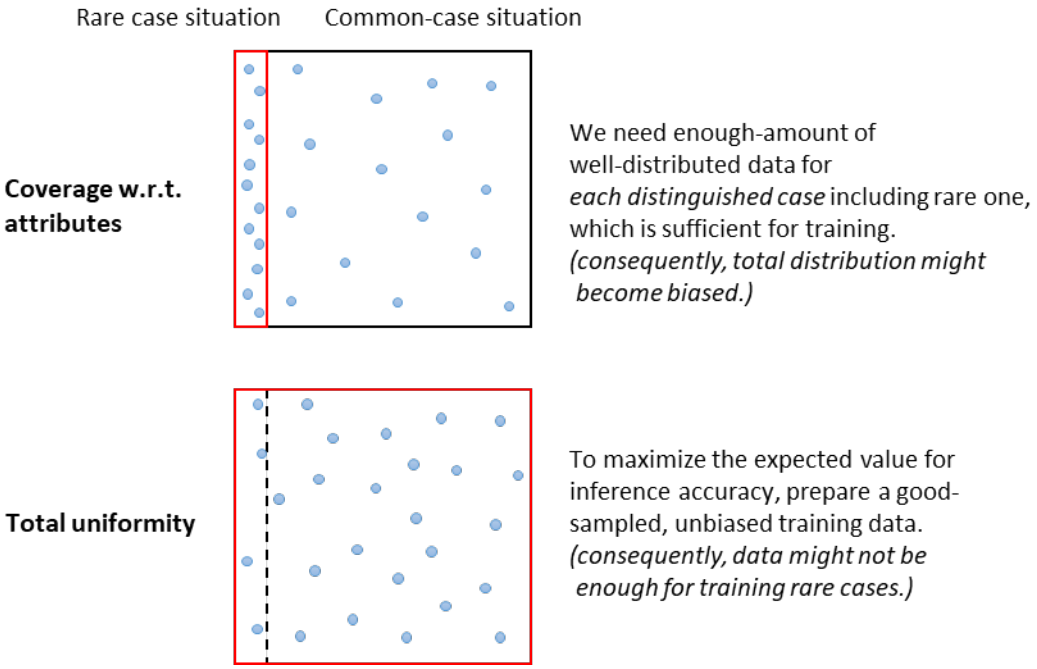

Figure 5: Relationship between coverage and uniformity

(Example 1)

For example, if snowfall may affect the performance of signal image recognition in an automatic vehicle, the necessary training or verification data must be prepared to suppress the effects of false recognition even in snowfall conditions that are expected to occur only one or two days a year in Tokyo, which is set as the operational area. For this reason, the proportion of snow images in the dataset may be increased compared to the actual probability of snow events.

In this case, it may be necessary to prioritize the *coverage of the dataset* over the *uniformity of the dataset*.

(Example 2)

On the other hand, if one or two snowy days a year are strongly trained in sales predictions of a retail store in Tokyo, the prediction performance in other climate conditions may

deteriorate and the average profit cannot be maximized. In this case, we might need to give priority to *uniformity of datasets*. Of course, what kind of data is actually prepared as training datasets will finally be examined in the implementation process taking the balance of both internal quality characteristics into consideration.

1.7.5. B-3: Adequacy of data

In contrast to the properties of the distribution of the data set in B-1 and B-2, the *adequacy of data* refers to the fact that each item of data in the data set is appropriate for the purpose of training. Adequacy includes not only the absence of errors in the values, but also the absence of data that should not be used for training (even if the values themselves are accurate) (consistency), the absence of inappropriate modifications to the data (authenticity), and the fact that the data is sufficiently new (up-to-date). In the context of supervised machine learning, two perspectives are also included: *adequacy of data selection*, which is the adequacy of the measurements to be trained (values that correspond to the input side when the machine learning element is viewed as a function), and *adequacy of labeling*, which is the adequacy of the correct answers added for training (values that correspond to the output side).

The quality of the adequacy of the data is primarily determined according to the requirements definition for the implementation of the machine learning elements. In general, the quality needs to be reevaluated when the requirements definition is updated, while there are some aspects that can be judged as general-purpose data (e.g. removal of obviously invalid data, authenticity and traceability of the data set as a whole). Even in the development process of a single system, when requirements definitions and labeling policies are updated due to trials in the PoC phase or regressions in the production phase, the data needs to be rearranged to accommodate the new requirements and policies, so it is important to carefully consider the man-hours and procedures in the development process.

Although this guideline aims to evaluate validity through inspection of data as much as possible, there are some aspects that are particularly difficult to check from the data itself, such as credibility and traceability, and some aspects, such as uniformity of labeling policies, that should be achieved through process management.

1.7.6. C-1: Correctness of trained models

The term *correctness of trained models* represents that a machine learning component functions as intended upon the input from the learning dataset (consisting of training data, validation data, and test data).

Normally, the training dataset does not provide sufficient information to reflect all input given from the in-operation environments. Therefore, a trained model might show high

performance with respect only to training data, but not to the data other than the training data. In other words, a trained model may overfit to the training data and may not generalize well to unseen data. When a machine learning component is evaluated using only the validation datasets and test datasets, it is not possible to check if the machine learning component functions as intended when given unseen data from the environment. Hence it is important to evaluate not only the correctness but also the stability, which will be explained in the next section.

1.7.7. C-2: Stability of trained models

The term *stability of trained models* represents that given an input that is not included in the learning dataset, a machine learning component behaves in an expected way. When a trained model does not satisfy stability, it may not function correctly upon the input data that are not included in the training, validation, or test datasets. For example, a trained model may function incorrectly when the input is perturbed by natural/adversarial noise.

1.7.8. D-1: Reliability of underlying software systems

The property *reliability of underlying software system* represents that the underlying conventional software (e.g., training programs and prediction/inference programs) functions correctly and reliably. This notion includes various software quality requirements such as correctness of algorithms, time/memory resource constraints, and software security as well.

1.7.9. E-1: Maintainability of quality in operation

The term *maintainability of quality in operation* means that internal qualities fulfilled at the time when the operation started is maintained throughout the operation period. The maintainability implies the following three aspects: internal qualities sufficiently adapt to changes in external environments; they do so also to changes in the system feeding input to the AI; and they do not deteriorate due to changes made in trained machine learning models to make such adaptations.

A specific method to realize quality maintenance depends on forms of operation, especially on how to carry out additional learning/iterative training. This point will be described in Section 6.9.1.

1.8. Concept of development process model

1.8.1. Relationship between iterative training and quality management lifecycle

Machine-learning based systems broadly apply more adaptative and flexible development processes such as the development of conventional waterfall process, the introduction of preliminary experiment stage called Proof of Concept (PoC) and agile development. Moreover, continuous development that realizes higher quality during operation and responds to environmental changes based on real data obtained in the operation stage is thought to be effective in machine learning.

Hence, the Guideline define not only early/late-stages of development in which software is actually built but also integrated stages including the stage of formulating system specifications and the actual operation stage as a system lifecycle process in which development and operation are integrated. The process of grasping quality goals in the problem domain analysis stage prior to system design and the processes of monitoring behaviors in operation, obtaining additional data and carrying out additional learning are positioned as one important stage for system development and treated as a part of initiatives for quality management.

The whole lifecycle is modeled as a hybrid process integrating agile (iterative) development process in which the progress is managed based on quality inspections (tests) and the whole top-down V-shaped development process focusing on process management (Figure 6).

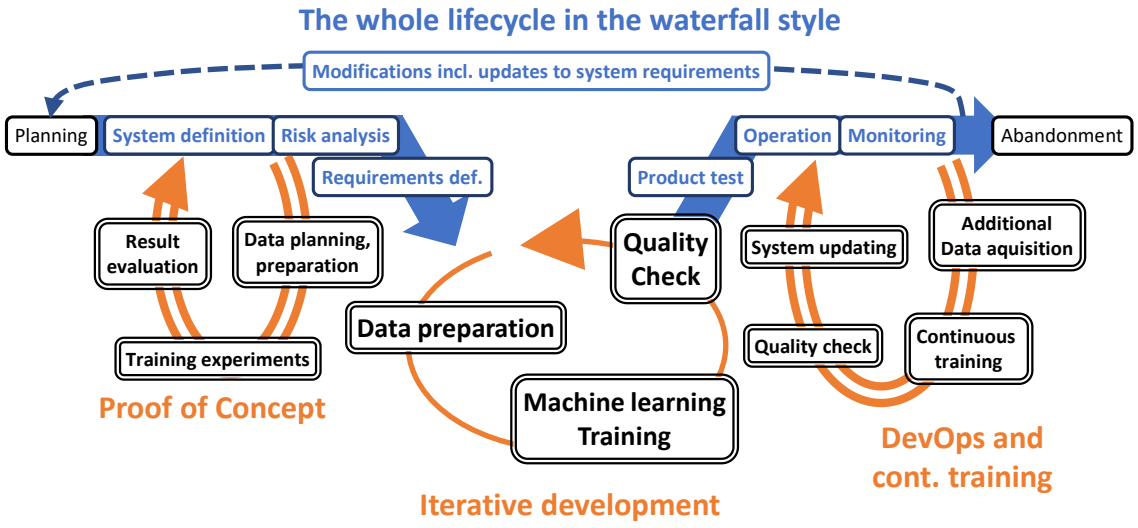

Figure 6: Conceptual diagram of mixed machine learning lifecycle process

Specifically, works to analyze requirements necessary for the whole system quality are

regarded as a flow process similar to a top-down process and assumed to be analyzed in detail as an independent process before actual system development and data reduction start. Reworking and iterative works at this stage suppose that the same level of consistency is ensured as final results even if works are restarted from the middle of the development process. Moreover, the process of obtaining additional data in operation and the monitoring process are positioned as a part of the iterative quality management process based on the concept of DevOps and should be compatible also to the concept of the top-down operation phase in RAMS specifications where necessary.

The stage of so-called PoC development which is seen often in the development of AI is categorized as a part of the process of the prior analysis stage leading to the definitions of requirements as an important quality management stage. This is a concept of identifying and categorizing requirements for quality again, when knowledge and data obtained from PoC are utilized, so that quality management in the main development stage and free explorative development in the PoC stage are balanced. In the actual development process, a trouble of re-categorization can be saved by partially introducing the concept of quality management of the main development stage in the PoC stage.

The lifecycle process in this guideline is a reference model, and development processes carried out by respective developers can be designed as their own. This reference model intends to help readers understand individual stages of the development process created in accordance with different circumstances in comparison to the Guideline and find out how quality management technologies included in each chapter of the Guideline correspond to development stages taken care thereby.

(Note)

The development process of AI has been often categorized as a non-waterfall iterative process. When iterative training is given, various works such as addition of collected data, change of selection and adjustment of parameters are carried out other than changes in algorithm implementation codes. Sometimes, the goals are achieved by modifying the principles for implementation and specifications based on those changes, giving training in accordance with new specifications and improving the accuracy. On the other hand, *gates of quality* are often established in the form of check before operation and inspections on orders, even if products that require quality adopt a circular development process model in reality.

A model shown in Figure 7 exemplifies, based on the development process of AI understood as a iterative system, its relation with the mixed process shown in Figure 6 which the Guideline basically assumes.

To be more specific, several development tests conducted until policies for implementation (also called specifications for implementation) are concretized are mapped to several circulations in the PoC stage to examine the quality. Then, one or several tests immediately prior to the operation stage leading toward training/quality tests and release based on the final specifications are positioned as *main development stage* in the mixed process.

At times, the specifications need to be modified again after going through the main development stage. From the perspective of waterfall process, this is a step back from the implementation stage to the requirement analysis stage, but from the perspective of iterate process, we do not need to consider it as a serious step back, because *it was found out that the product was still in the PoC stage*. The PoC development stage has an aspect of implementation forecast and knowledge obtained in this stage is reflected implicitly in the main development stage. Therefore, even if this process is considered as a waterfall type, efforts for implementation in the PoC development stage are not in vain.

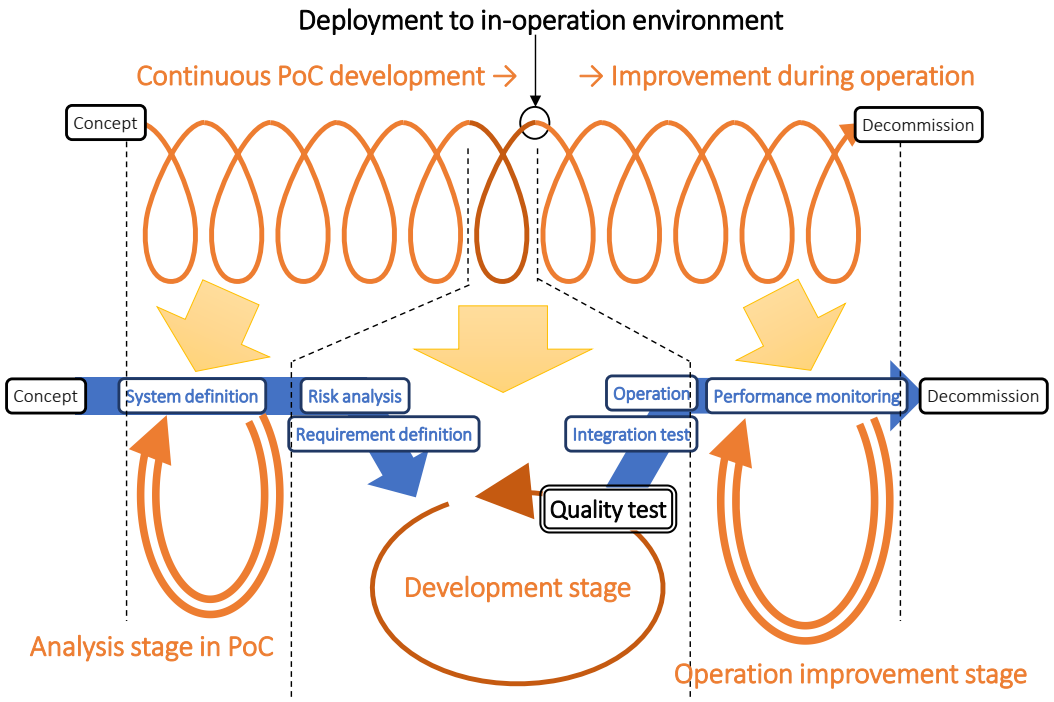

Figure 7: Relationship between the circular process modeling and the model in the Guideline (example)

(Note 2) Section 4.1.1 (Page 46) describes changes in operational situations and the concept of operation with various gates as an example of application.

1.8.2. development process by multiple stakeholders

In the actual development of AI, works may be divided in various forms. For example, a service provider itself may take care of planning, development, and operation, outsource only the training stage or entrust system design and build-out of a trained model and incorporate it into systems. In the quality management process defined in the Guideline, a development

entruster responsible for product planning plays a central role and quality management activities are carried out by everyone including developers, etc. Management works for individual stages in the process are shared upon agreement between workers. As a result, a development entruster or development entrustee may take responsibility for setting qualities in use and external qualities based on the purpose of applied systems depending on the form of division[4]. In any case, it is important to reach agreement to ensure sufficient quality for end users together, and clearly reflect the share of costs in the contract so that activities in the quality management process that continue until the operation starts are not disrupted.

Sections 5.2 and 5.3 explains some sharing models and remarks when several entities involve in development such as consignment development and delta development.

1.9. Relations with other documents and rules

1.9.1. "Social Principles on Human-centric AI"

In Japan, the Cabinet Office has defined an ideal relationship between operators and developers of machine-learning based systems and the society and general public in the form of "Social Principles on Human-centric AI" [20].

In relation to those Principles, the Guideline are primarily positioned as non-binding guidance which sorts out technical matters developers themselves refer to and practice in order to improve the quality and prevent safety and security from being compromised, when operators and developers fully understand those Principles and actually develop machine-learning based systems and machine learning components therein (Figure 8). Moreover, the Guideline are, together with other guidelines and future international standards, also positioned as an element that constitutes "Social Principles of Human-centric AI" mentioned therein.

[4] In cases where a requirement for validation that *certain performance indicators (accuracy, etc.) should be achieved on data given by a development entruster* is established as a form of order requirements common particularly in the development of AI, we need to take note that the development entruster is responsible for ensuring so-called "data quality" (*the data is appropriate for learning consistent with the purpose of products*) as explained in 1.7.1 to 1.7.4 of the Guideline. In cases where the development entruster is not capable of validating or ensuring data quality, it is necessary to explicitly entrust a part of the work in the form of *design support work* and consider a possibility that the system cannot be built due to the lack of data quality when it is actually trained (the work is returned to the development entruster).

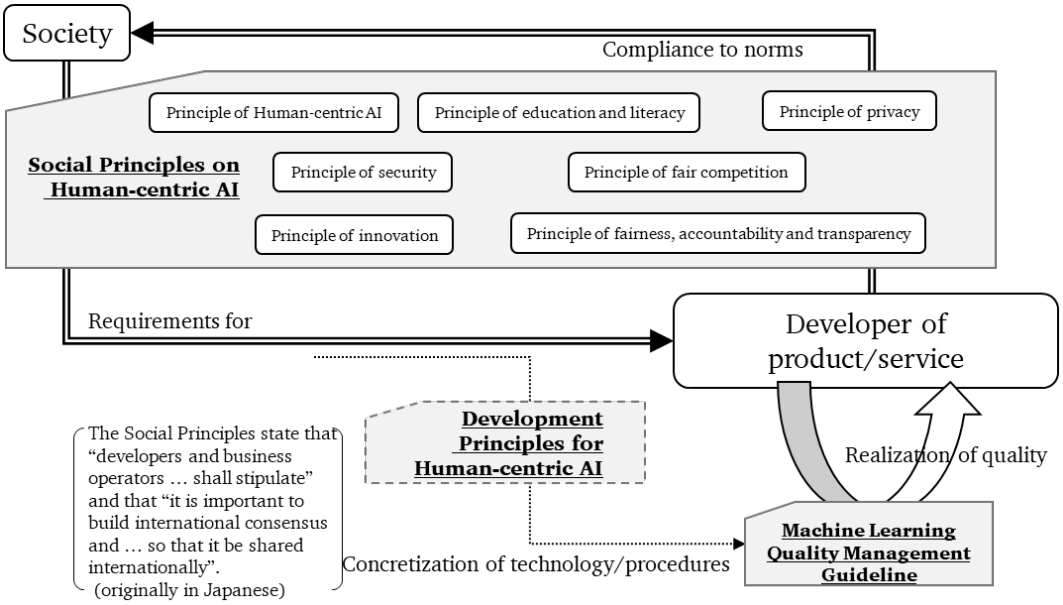

Figure 8: Relation with "Social Principles on Human-centric AI"

1.9.2. AI-related rules and guidelines of foreign governments and international organizations concerning AI technology

In addition to the above principles, norms for social nature, safety and ethics concerning the development and use of AI technology have been documented recently in different forms [21][26]. As for their relationship with the Guideline, those social norms are positioned under verbalized documents as in the case of the "Social Principles of Human-centric AI" mentioned in the previous section and categorized to present one specific method for realizing some of them in system development.

1.10. Structure of the Guideline

The structure of the remaining parts of the Guideline is as follows.

- Chapter 2 sorts out the scope of the Guideline and their relationship with existing standards again.
- Chapter 3 sorts out the details about qualities in use mentioned in Section 1.5 including decisions on leveling.
- Chapter 4 shows reference models with regard to the development processes whose outline was explained in Section 1.8.
- Chapter 5 mentions specific methods of operating and applying the Guideline.
- Chapter 6 sorts out the internal quality characteristics mentioned in Section 1.7 in

more detail.

- Chapter 7 sorts out possible ways of realizing each internal quality characteristic described in Chapter 6.
- Chapter 8 details issues pertaining to the management of fairness, one of the external qualities.
- Chapter 9 elaborates on approaches to security issues in the machine learning quality management.
- Chapter 10 shows reference information such as the relationship with other guidelines.
- Chapter 11 shows the process of analysis and concept until the Review Committee of the Guideline clarifies the internal qualities listed in Section 1.7 (and Chapter 6) as reference information.

(Note)
The Guideline can be read in the following ways (other than reading in order from the beginning).

Read Chapter 5 to understand the outline of the procedures for processes.

Read Chapter 4 to determine the quality level referred to in the development process.

Refer to each section of Chapter 6 (and the table in Section 12.1) to clarify check items required at each level.

Refer to each section of Chapter 7 as needed to grasp specific concepts of the above check items and applicable technology.

The definitions of the development stage, etc. referred to in each section are summarized in Chapter 4.

2. Overview

2.1. Scope of the Guideline

2.1.1. Products and systems subject to the Guideline

Products and services subject to quality management in this Guideline are those that use machine learning technology for the building of some of their software components among all information-processing systems such as industrial products, consumer equipment and information services (hereinafter referred to as *machine-learning based system* in this Guideline).

This Guideline mainly considers about machine-learning components implemented in supervised learning although the basic approach can be also applied to other methods of implementation such as unsupervised learning, semi-supervised learning and reinforcement learning. Future revisions of this document will consider specific methods of handling for them.

2.1.2. Products and services subject to quality management

The quality characteristics which this guideline set goals, manage, guarantee is the external quality corresponding to the effect on the system which implements the software component built with the machine learning technology (hereinafter referred to as *machine learning component*) that is the internal component constructing the machine learning based system.

On the other hand, quality management mentioned in this Guideline directly target internal qualities machine-learning components have.

We consider that qualities in use of the whole system are realized comprehensively by external qualities of elementary parts of that system. Moreover, external qualities of each component depend on the quality of internal components/parts and internal quality characteristics which are the internal quality of those components themselves. Quality management is realized by sorting out the requirements for machine-learning components as internal quality characteristics and by managing the process toward the achievement thereof (Figure 9). However, among system components, software and hardware other than machine-learning components are explained only within the necessary scope of their relationship to this Guideline for the purpose thereof.

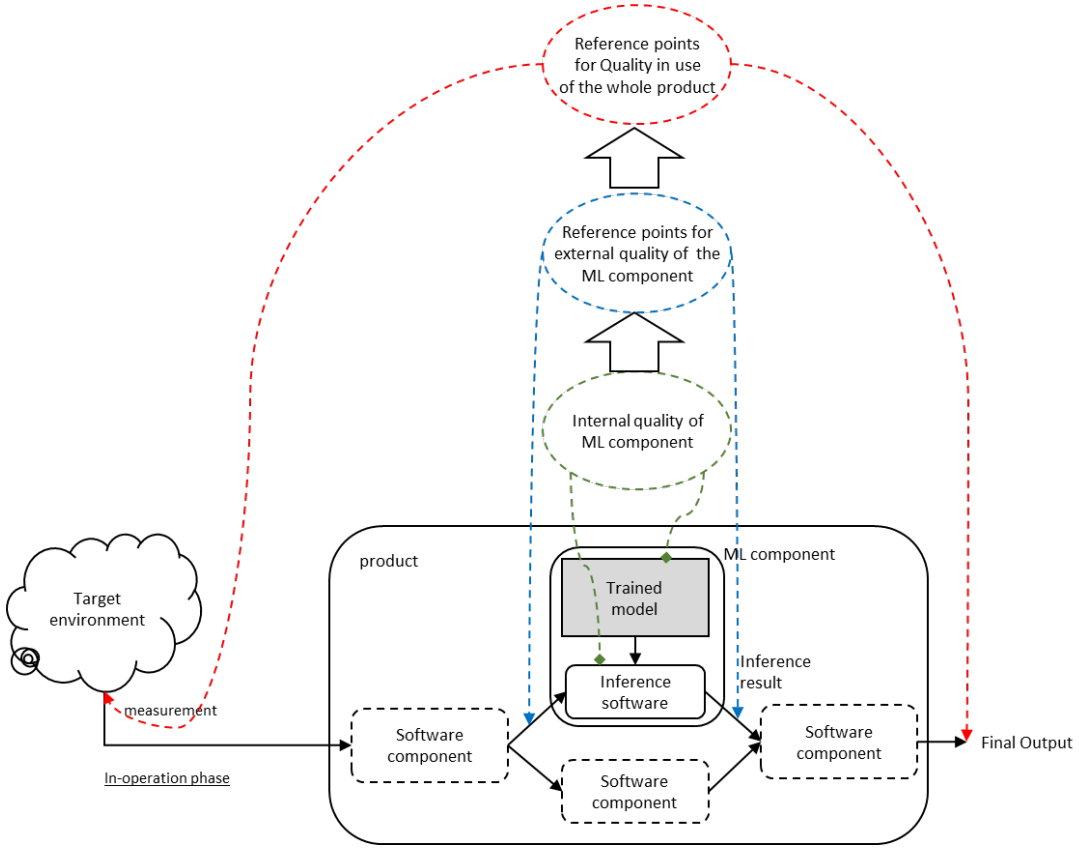

Figure 9: Concept of quality requirement of whole AI-based system and means for realizing them

2.1.3. Scope of quality management

In the Guideline, the term *quality management* refers to a process of quality activities across the lifecycle from the planning stage to the operational stage of machine-learning based systems and includes such meanings as setting of quality goals, planning, confirmation, quality assurance and management. Although the meaning of *quality management* is wider than quality assurance or management of general software, it does not include organization planning and management of responsibilities or resources which are included in the meaning of quality management defined in ISO9001.

2.2. Relationship between this Guideline and the other standards for system quality

Mainly, this part sorts out the relationship between the Guideline and the existing international standards for information system quality. Also See Section 1.9 for their relationship with guidelines equivalent to superordinate concept such as sociality.

The Guideline present quality management methods for a variety of applications of

machine learning and define systems that require safety as complement or extension of some of conventional standards for functional safety (IEC 61508-3 [13], IEC61508-4 [14], etc.).

- For systems that strongly require functional safety, functional safety standard IEC61508-1 [12] or other equivalent standards are primarily applied.
- For ensuring the required safety level of the machine-learning component in the targeted system, this guideline sorts out the issues and methods for ensuring safety that is unique to the machine-learning and propose the methodology for complementing or replacing the methods introduced in IEC 61508-3 comparing with the conventional software.

2.2.1. Security standard ISO/IEC 15408

The Guideline and information system security standards such as ISO/IEC 15408 [3] are independent, and they should be applied simultaneously when needed. Information security is a mandatory requirement for systems that require integrity and availability in order to realize risk avoidance included in the Guideline, but the basic countermeasures defined in those standards are also applicable to machine-learning based systems.

2.2.2. Software quality model ISO/IEC 25000 series

The ISO/IEC series which define software quality models, especially ISO/IEC 25010 [7], can be partially applicable to machine-learning based systems.

Since product quality concerning software components are analyzed from the viewpoint of conventional software, quality properties sorted out in the above standards are supposed to be achieved also in machine-learning components. However, since there are differences from product quality and *analysis and decomposition into components* focused in the Guideline, we assume that there is no clear relationship between them. Although there is a proposal to improve the above standards to machine learning and AI at an international level, we continue to discuss this matter in view of future standardization.

2.3. Definitions of terms

This section shows definitions of terms used in this guideline. Some of the terms on artificial intelligence and machine-learning are subjects of discussions in ISO/IEC 22989 [4], currently in the DIS stage. Definitions of those terms in this guideline are to be aligned after the discussions conclude.

2.3.1. Terms related to machine-learning based system structure

1. Machine learning based systems / systems using machine learning technology

A system containing software components (machine learning components, 2.3.1.2) implemented by applying machine learning technology.

(Note) The common terminology “machine learning systems” refers to machine-learning based systems or machine learning components (2.3.1.2) depending on the context.

2. Machine learning component

A software component implemented by applying machine learning technology. A machine learning component realize the functions of trained machine learning models (2.3.1.5) as software. Generally, this component consists of prediction/inference software component (2.3.1.6) implemented as software and trained machine learning model (2.3.1.5) incorporated as fixed input.

3. Machine learning algorithms

An algorithm that provides the calculation method for prediction/inference of machine learning and for obtaining said calculation method through training. Each of the calculation methods correspond to prediction/inference software component (2.3.1.6) and trained machine learning model (2.3.1.7).

There are a lot of types of machine learning algorithms, such as neural networks, support vector machines and decision trees and so on. Appropriate algorithms are selected based on the type and purpose of knowledge which machine learning components are to obtain.

4. Hyperparameter

A setting value input in training software component (2.3.1.7) to execute machine learning training. It may need to be adjusted in accordance with the progress of training. In some cases, a hyperparameter may not exist, adjustment of hyperparameter may be omitted intentionally, or a technique to adjust it automatically may be used.

(Note) As regards the diversity of mathematical structure that can be subject to machine learning, which scope is fixed as *machine learning algorithms* prior to the start of training and which scope is set/adjusted during training as *hyperparameters* are arbitral depending on how to implement software and selection of policies for training by developers. In some cases, the structure of calculation formula itself is treated as data and subject to automatic adjustment for optimization as a part of hyperparameters.

5. Trained model / trained machine learning model / knowledge base / trained parameter

Information required as output for training that defines functional operations of machine learning.

(Note 1) *Machine learning model* in the Guideline refers to trained machine learning

model or in-processing data for obtaining said model.

(Note 2) What is simply called “parameter” in the context of machine learning technology often refers to this trained machine learning model or numerical data included therein.

(Note 3) Mathematical structures such as neural networks and Bayesian networks are sometimes called as “graphical model” or “statistical model”. Accordingly, the terms such as “selection of machine learning model”, “model design” and “untrained model” may be used for the selection of machine learning algorithms and the setting of their parameters.

(Note 4) In literatures, the term “trained model” is used in comparison to “trained parameter” either as *output of model training* or as *trained machine learning model implemented as specific software*. In this Guideline, the term *trained model* is used for the former meaning while the latter is called *machine learning component*, to clearly distinguish output from training itself and pre-implemented prediction/inference software component.

(Note 5) When the context on machine learning or AI is clear, there may be no problem in calling it simply as “trained model”.

6. Prediction/inference software component

A part of machine learning components used during operation which is fixed as the implementation of software of machine learning models. This component receives a trained machine learning model (2.3.1.5) as static (semi-static) input and data obtained from real environments as dynamic input.

7. Training software component

A component to generate trained machine learning models (2.3.1.5) using training datasets. This component implicitly corresponds to prediction/inference software component (2.3.1.6) as the implementation based on different aspects of the same machine learning algorithms so that they are usually used as a pair.

Although a training software component is not included in machine learning components used during operation in many cases, there are the cases (e.g. a system that runs the online learning during operation and reinforcement learning) where the training software component is included as a part.

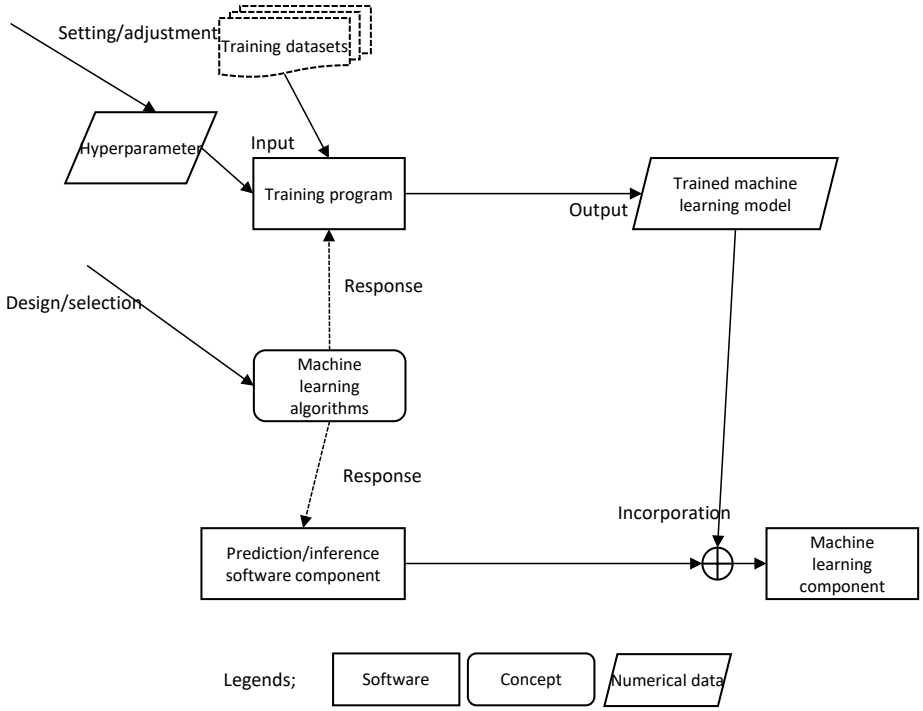

Figure 10: Relationship between terms concerning the structure of machine learning based systems

2.3.2. Terms related to stakeholders of development and their roles

1. Service provider

An entity that uses machine learning based systems for its own purpose or for selling or providing services for customers. Moreover, as regards a business structure in which software providers, etc. plan and develop in advance systems in anticipation of services provided or used by others and sell them as packaged or customized systems, those entities that engage in planning and development are treated according to the service providers.

2. Self-development entity

A product development entity designs and implements the machine learning components.

3. Development entruster

A product development entity that asks others to implement machine learning components regardless of the form of contract (service/entrustment).

4. Development entrustee

An entity that implements machine learning components upon request from the development entruster.

5. (Development) stakeholders

All stakeholders that engage in development and operation of machine learning based systems, including production operators, self-development entities, development entrusters and development entrustees.

2.3.3. Terms related to quality

1. Harm avoidance (Risk avoidance/safety)

A quality characteristic of avoiding negative effects such as human damage, economic loss and opportunity loss on operators, users of products or third parties caused by undesirable judgements of machine learning components. The improvement of *risk avoidance* corresponds to a concept of *risk reduction* in the safety field.
[Defined in 1.5.1 and 3.1 in the Guideline]

2. AI performance

This characteristic allows machine learning components to output expected by machine learning based systems and their users at a higher precision/probability than the average in the long run. AI performance is evaluated based on comprehensive performance rather than whether the output is acceptable or not. (Defined in Sections 1.5.2 and 3.2 in this Guideline.)

3. Fairness

The output or distribution of machine learning components is not affected by differences in some of the attributes belong to the people or others (or the effect is kept sufficiently low).
(See Sections 1.5.3 and 3.3 of this Guideline)

4. Attack resistance

This characteristic prevents machine learning components (or machine learning based systems) from reacting contrary to the operator's expectations (in line with an attacker's intention) in relation to input data or external environments built intentionally by the attacker.

5. Ethicalness

The behavior of machine learning based systems is appropriate in the human-centered society.

6. Robustness

This characteristic allows systems to maintain their performance level under any circumstance.

2.3.4. Terms related to development process

1. System lifecycle process

A process management model looking down the flow from the planning of systems to the end of operation and disposal.

[Source of definition: ISO/IEC/IEEE 15288 [2]]

2. Agile development process

A collective term of agile and value-driven development approaches.

(Note) This term originates from the Agile Manifest announced in 2001 and can be utilized in various forms such as scrum and XP.

3. PoC (Proof of concept)

Preliminary development activities carried out with the aim of validating feasibility of ideas to be realized and solution for the problem instead of being realized as products.

4. Systems engineering

An approach across several fields to deploy balanced system solutions in response to diversified needs of stakeholders. It applies both management process and technological process and strikes a balance between them to reduce risks that affect the success in projects.

5. RAMS (Specification and demonstration of Reliability, Availability, Maintainability and Safety)

In the Guideline, RAMS refers to a concept of the comprehensive system lifecycle process specified in the standard IEC 62278 [15] (EN 50126) about the reliability management process in the field of railway service.

(Note) *The RAMS standards* may refer generally to IEC 62278 itself.

2.3.5. Terms related to use environment

1. Environment

In the Guideline, this term is used in three contexts: 1) External environment, 2) computing environments, and 3) operational environment.

2. External environment

The physical environment or cyberspace to be the source of input to the machine learning based system, and there is interference from the environment to the systems or from the systems to the environment.

3. Open environment

The external environment whose system operation is greatly affected by the condition

of the people other than users or the things such as nature.

4. Environmental condition

A characteristic which distinguishes the differences in conditions of external environment. From the viewpoint of machine learning quality management, we focus especially on changing characteristics such as the status of hazard and risks.

5. Computing environment

Software execution environment where machine learning components (prediction/inference software components) and training software components are executed. It may include hardware environment, operating systems and middleware on which computing environment is based depending on situations.

6. Operational environment

Operational environment includes computing environment and operational structure of humans.

7. In-operation environment

Production environment where machine learning based systems are provided.

8. Runtime (computing) environment

Computing environment of computers, clouds and IoT devices where machine learning based systems including machine learning components are operated in in-operation environment.

9. Development environment

Computing environment where machine learning components are built and modified and such modifications do not affect actual operation directly, and operational environment including computing environment thereof.

2.3.6. Terms related to data used for building machine learning

1. Training dataset

A set of data used for training of machine learning models in the iterative training phase.

2. Training data (instance)

Data included in training datasets.

(Note) Generally, the term “data” is used for both single instance and several instances (a set of instances). In this Guideline, the term *data* is used only for the former, while the

term *dataset* is used for the latter, in order to avoid this ambiguity. That is, the term *dataset* is used for the case that it is treated as one set which can be the target of discussion about the distribution and the total quantity of a set. On the other hand, the term *data* is used to refer to individual data points or discrete data points before it is captured as a block. The relationship between these terms is described by expressions of the relationship between set and element such as “added to dataset” and “included in dataset”.

3. Validation dataset

A set of data used for evaluating convergence of machine learning models in the iterative training phase.

4. Test dataset

An set of data used by machine learning models built as input in tests as one or more means for checking the desired quality and performance in the quality check/assurance phase.

5. Adversarial example(s)

Data built with intention so that the inference results to be different from assumptions/intuition are output when it is input into the machine learning components.

6. Overfitting

Trained machine learning models are adapted excessively to training data and it becomes impossible for them to return desired output in response to input data other than training data.

7. Attribute

Each itemized element to analyze and classify the characteristics of environmental conditions in ML problem domain analysis.

8. Value

Types of specific characteristics of environment included in each attribute classified in the ML problem domain analysis.

9. Labels

Identifiers belonging to data used for classification problems in supervised learning, indicating correct classes which they belong to.

2.3.7. Other terms

1. Software regression

In software engineering, this term means that, when software is improved, it stops to operate as expected in response to input which did not cause problem before.

(Note) In machine learning and statistical analysis, the term *regression* is often used as *regression analysis*.

2. SIL (Safety Integrity Level)

The classification of levels related to the achievement of functional safety of products as specified in IEC 61508 [Source of definition: IEC 61508-1]

3. KPI (Key Performance Indicator)

An indicator which quantifies the attainment level of functional requirements to be attained by output from machine learning components through machine learning based systems.

4. Additional training

A form of operation to collect data and carry out additional training for machine learning during operation and to update trained machine learning models when necessary.

(Note) This is a concept including not only *online learning* described below but also *additional offline learning*.

5. Online learning

A form of system implementation in which additional training is carried out without going through the validation process in development environment and its results are reflected in operation environment.

(Note 1) In this Guideline, a form of carrying out additional learning in development environment and updating model parameters by such means as firmware updates after going through the assurance process without online learning is called *additional offline learning*.

(Note 2) The term *continuous learning* seems to be used as a synonym for either *additional training* or *online learning* here.

3. Levels for external quality characteristics

As described in Section 1.5 of this Guideline, this guideline defines three aspects of external qualities for machine learning components contained in machine learning based systems. In this section, we set target levels of quality for each aspect. The procedures for determining target levels in development will be provided in Section 5.1.2.

3.1. Safety / Risk avoidance

For the safety or risk avoidance aspects, quality levels are classified into seven *AI safety levels* (AISL 4, AISL 3, AISL 2, AISL 1, AISL 0.2, AISL 0.1, AISL 0). Throughout this guideline, causes of the *safety* risks include both physical hazards such as injury of humans and economic hazards as well.

The required AISL shall be determined using Table 1 and Table 2, depending on the nature of corresponding hazards[5]. If the cell containing the "*" in Table 1 is referred, *the functional safety standard IEC 61508-1* [12] *or any similar standard for specified application fields is consulted* first, safety function is assigned for the components in the system, and the SIL 4–1 assigned to the machine learning component in question (or any equivalent assignments in other related standards) is translated to AISL 4–1 respectively. Optionally, even in the other cases, stakeholders can apply risk analysis based on IEC 61508-1 or others and assign AISL based on the SIL evaluation result.

In the meantime, this Guideline does not expect AISL 4 or 3 is directly assigned to any machine learning component. In that case, the overall system design should be reconsidered to reduce risks caused by the machine learning components in question.

Table 1: Estimation of AI safety levels for human-related risks

Expected severity of hazards/possibility of avoidance	Impossible to avoid	Possible to avoid through monitoring by humans	Always requires check/manual operation by humans
Simultaneous deaths of multiple persons	*	*	*
Death or injury of one person	*	*	*
Injury with left disability	*	AISL 2	AISL 1

[5] Each AISL is described to correspond roughly to safety integrity levels 4~1 of the standard IEC 61508 (functional safety). Furthermore, to precisely recognize safety requirement strength to be categorized as *No SIL applicable*, the corresponding AISLs are divided into three classes as 0.2, 0.1 and 0.

Serious injury	*	AISL 1	AISL 0.2
Minor injury	AISL 1	AISL 0.2	AISL 0.1
Minor injury (where possible victims can avoid the hazard easily)	AISL 0.2	AISL 0.2	AISL 0.1
No damage is expected	AISL 0	AISL 0	AISL 0

Table 2: Estimation of AI safety levels for economic risks

Expected severity of hazards/possibility of avoidance	Impossible to avoid	Possible to avoid through monitoring by humans	Always requires check/manual operation by humans
Damages affecting continuity of business entities	(AISL 4)	(AISL 3)	AISL 2
Serious damage that undermines business operations	(AISL 3)	AISL 2	AISL 1
Considerable/specific damage	AISL 2	AISL 1	AISL 0.2
Minor loss of profit	AISL 1	AISL 0.2	AISL 0.1
No damage is expected	AISL 0.1	AISL 0	AISL 0

(Note) AISL 0.1, 0.2, 1, 2, 3, 4 may be considered to correspond to Sensor Performance Classes A to F in IEC 62998 [16] respectively. AISL 0.1 to 3 may also be considered to correspond to Performance Levels PL_a to PL_e in ISO 13849 [1] respectively.

3.2. AI performance

For the AI performance aspects, quality levels of components are assigned by the agreement between stakeholders based on the following criteria.

- AIPL 2 (mandatory requirements):
  - When it is indispensable or strongly expected that the product or service satisfies certain performance indicators (e.g. accuracy, precision or recall) for the system's operation.
  - When the fulfillment of the certain performance indicators is clear stated as requirement in contracts.

- AIPL 1 (best-effort requirements):
  - When certain performance indicators are identified as an factor for satisfying purpose of the product, but AIPL 2 is not applicable. For example, when shorter development period is beneficial, or when gradual improvement of performance to a satisfactory level during in-operation phase is acceptable.
- AIPL 0
  - When performance indicators are not specified at the beginning of development, and discovery of a performance indicator itself is the purpose of development.
  - When the development completes at the Proof of Concept stage.

3.3. Fairness

Quality levels of components for fairness aspects are determined based on the following criteria.

- AIFL 2 (mandatory requirements)
  - When a certain level of fair treatment is required according to the laws, regulation or equivalent de-facto social guidelines.
  - When the product deals with personal data and its output directly affects right of individuals.
- AIFL 1 (best-effort requirements)
  - When it is possible to define the requirements that there is no bias into the product or the services.
  - When there is possible demand for the explanation of the fair treatment provided by machine learning component (or AI in general) that might affect social acceptability or operation of the product.
- AIFL 0
  - When there are no identifiable requirements for fairness of the product or service.
  - When the product or service would be affirmatively accepted even if their outputs are unfair or non-uniform, in consideration with other factors such as performance.
  - When requirement for fairness is understood as *high performance regardless of input class*. In this case, it will be interpreted as a requirement for diversity support of input classes, related to other properties (risk avoidance or performance).

    (For example, in an application to detect some mechanical defects, if a detection rate for some specific defect class is relatively high, there are no need to *decrease* the detection rate to match with those for other unfavored classes).

4. Reference models for development processes

This Chapter explains the development process reference models for machine learning based systems which the Guideline refer to on the premise of discussions.

The reference models mentioned in this Chapter do not force developers to adopt a certain development method of machine learning based systems. Rather, they aim to add referable names to internal components and development processes of general machine learning based systems and compare recommendations provided for in the Guideline with actual development processes. By this way, the respective unique configurations and development processes can be compared with this model.

As shown in Figure 6 (page 25), this Guideline roughly divides the development process of machine learning components (and the principal parts of machine learning based systems that contain them) into three phases.

4.1. PoC trial phase

PoC trial phase as a development lifecycle process is, in the earliest stage of system development, sorted out as the preparatory stage to identify a possibility of performance attained by a system, quality degradation risks and its possible cause in advance of defining the functions of the whole process. Quality management activities in this phase are *sorted out as important preparatory works for quality management activities in the main development phase* in the context of quality management defined in the Guideline. From the viewpoint of data science, analytical activities in this phase are closely associated with subsequent risk analysis and requirements analysis and their results are often used in the subsequent main development phase. In order to attain *sufficiency of requirements analysis* (Section 6.1), it is essential to make efforts in the PoC trial phase. This Guideline stipulate that the adequacy of those activities is checked again in each step of the main development phase ultimately. This allows quality management activities to be conducted on a trial-and-error basis in the PoC phase without restrictions.

4.1.1. Handling of PoC phase including trial operation

Moreover, *Proof of Concept* does not only mean simple data collection and analysis, trial training, and build-out of models but includes what is tested in an in-operation environment, although final products will be used in different situations (for example, under monitoring by humans). Furthermore, the operation stage may be divided so that quality requirements continuously shift to different operational conditions. The Guideline consider such type of

development with physical operation divided in several stages as equivalent to the whole development lifecycle including the *main development phase* in relation to the system building stage in each stage and treat all achievements in the early operational stage as knowledge obtained from the PoC stage in later stages of development (Figure 11).

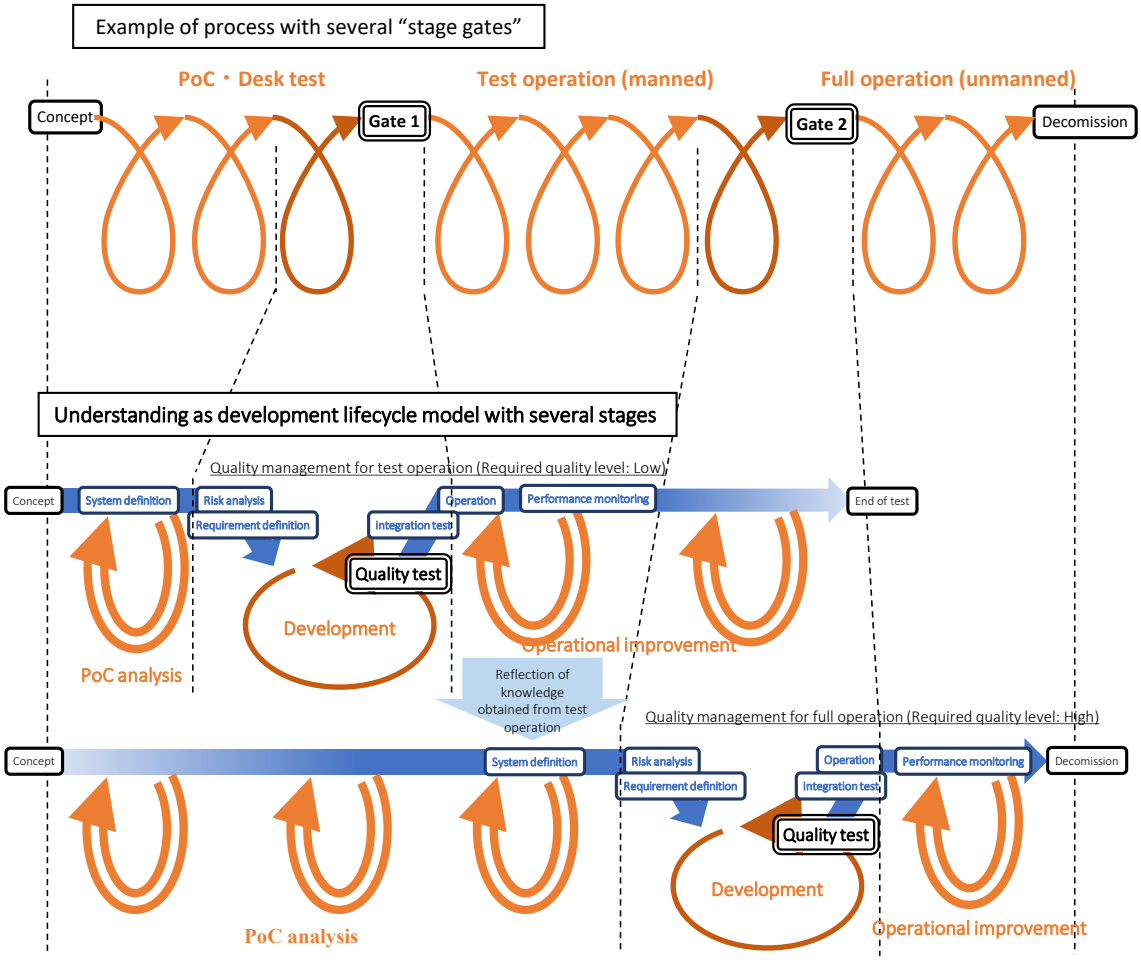

Figure 11: Handling of development process with several operational stages (Example)

4.2. Main development phase

Main development phase is a process from when system’s functional requirements are confirmed by means of the efforts in the PoC trial phase to immediately prior to the in-operation phase. This phase includes system definition equivalent to very early stages of development of conventional systems and integrated tests prior to final operation and shipment.

In the main development phase, a lifecycle which mixes a part of conventional waterfall early-stages of development and iterative development processes specific to machine learning

components is defined as a model. To be more specific, system requirements for each component including functional definitions and risks analysis of the overall system, analysis and decomposition of effects of components and machine learning components are decided. A series of processes which sets the goals for the last integrated tests prior to shipment are sorted out pursuant to the conventional waterfall model. Moreover, components of software and hardware other than machine learning components are assumed to adopt the conventional waterfall development model or agile development process[6] capable of ensuring the same level of quality as needed in late-stages of development. On the other hand, a development process of specific machine learning model equivalent to *late-stages of development* of machine learning components defines the iterative development process model again in the following section.

4.2.1. Machine learning model building phase

After risks which machine learning components have to deal with as well as quality requirements are specified in the main development phase, a machine learning model is built and its external qualities are tested (using indicators for internal qualities) and this process is repeated until the model passes a test. This is called *machine learning model building phase*.

A model which breaks down this phase into more specific process is shown in Figure 12. Each process of this model aims to correspond it to the descriptions in the Guideline by comparing a development process specific to each developer with this model so that it is not to restrain the development process unambiguously.

[6] *Agile development process* here means test-driven development or another non-waterfall style of development process which includes quality assurance activities comparable to those of waterfall processes so that expected levels of quality is reasonably explainable. It does not include all forms of non-waterfall development.

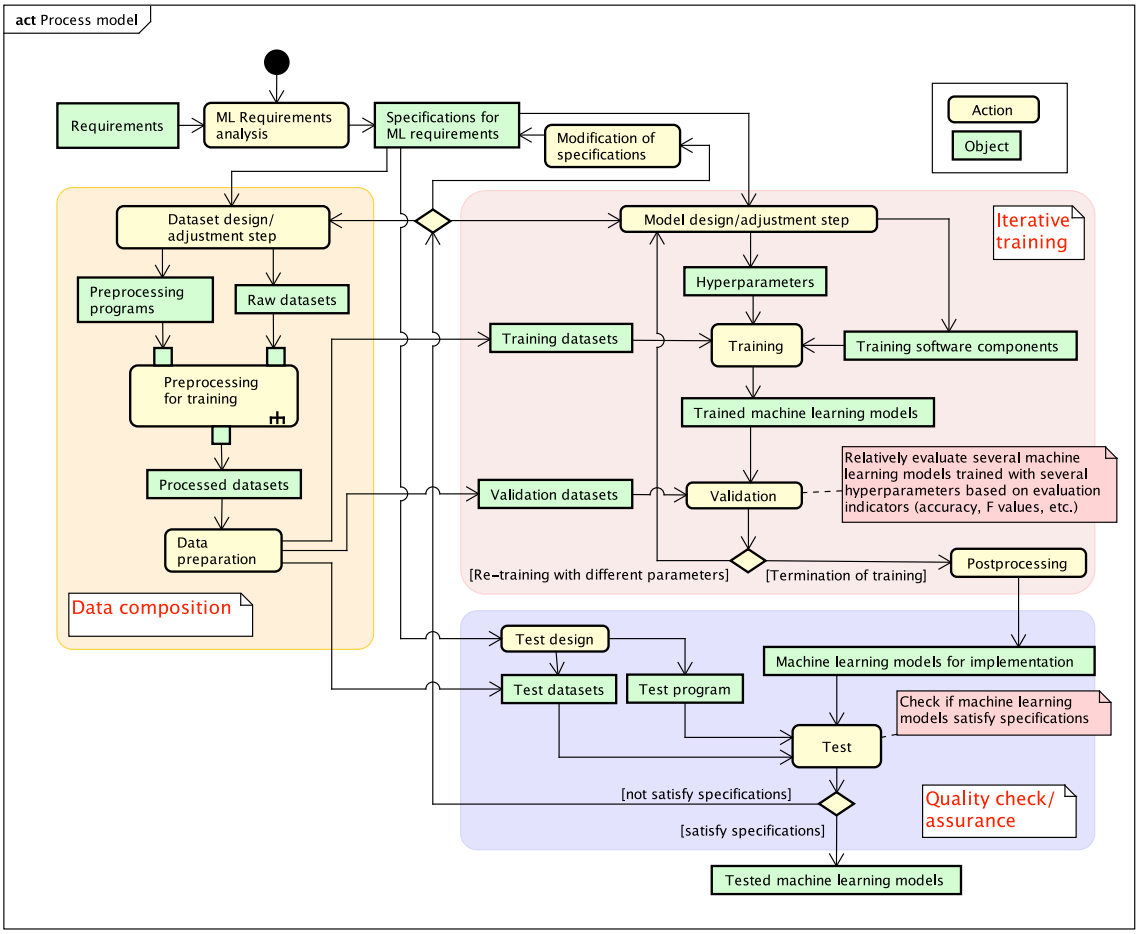

Figure 12: Process model in the stage of machine learning building

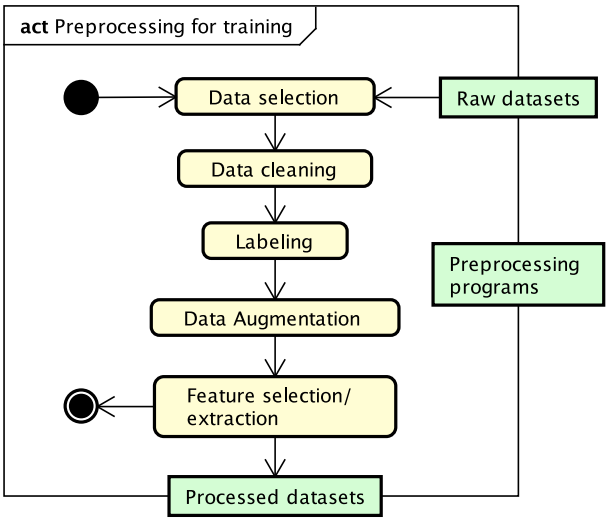

Figure 13: Example model of preprocessing for training

4.2.1.1. ML requirements analysis phase

In the *machine learning requirements analysis phase*, conceptual quality requirements for machine learning components are concretized by sorting out characteristics of data input to or output from machine learning components and re-organized as specific requirements for build data in the ML model building phase. In the actual development of machine learning based systems, performance requirements (quality requirements) are specified in many cases based on data characteristics or specific datasets. By explicitly including this analysis phase, it becomes possible to sort out a specific method of managing data quality and the relationship with other components in terms of quality management. This phase is closely related to internal qualities mentioned in Section 6.1 and Section 6.2.

4.2.1.2. Training data composition phase

In the training data composition phase, training, validation, or test datasets are generated in accordance with specifications for ML requirements. This phase is explained below in line with *composition of training data* shown in the upper left side of Figure 12.

4.2.1.2.1. Dataset design/adjustment step

In the dataset design/adjustment step, sufficient data is collected for each case in accordance with the specifications for ML requirements and design is carried out to generate datasets suitable for training. Output of this step is unprocessed datasets and preprocessing programs. In cases where it is found that trained machine learning models do not satisfy the specifications after going through training, the process may return to this step to adjust the design of datasets.

4.2.1.2.2. Preprocessing step for training

In the preprocessing step for training, raw datasets generated in the dataset design/adjustment step in Section 4.2.1.2.1 are processed into datasets suitable for training. As Figure 13 shows, there are cases where each process is performed in random order, in parallel, or repeatedly.

4.2.1.2.2.1. Data selection

Datasets used for training and evaluations (validation and tests) are selected taking into account coverage and uniformity from a large quantity of raw datasets. Since coverage and uniformity are traded off, the balance between them should follow the specifications for ML requirements.

4.2.1.2.2.2. Data cleaning

Noise removal, data forming, complement of missing values and removal of outliers are carried out so that original characteristics of the datasets can be trained.

4.2.1.2.2.3. Data Augmentation

In order to ensure the amount of training data or test data and to prevent overfitting, data variants are generated and added to datasets. For example, controls such as rotation, scale-down, reverse, change in luminosity and background replacement of image data are carried out to generate variants.

4.2.1.2.2.4. Label addition

Labels for supervised learning that show correct answers suitable to achieve requirements are added to the dataset.

4.2.1.2.2.5. Feature selection / extraction

Feature extraction is performed to add new features useful for model training, and feature selection is performed to reduce unnecessary features. For example, feature extraction uses techniques such as calculating frequency components by Fourier transform and finding the principal components of a plurality of features. If useful features are obtained, in many cases more unnecessary features can be reduced by feature selection.

4.2.1.2.3. Data preparation step

In the data preparation step, processed datasets generated are divided into training, validation and test datasets. Training datasets and validation datasets may be replaced during iterative training.

4.2.1.3. Iterative training phase

In the iterative training phase, machine learning models are designed in accordance with the specifications for ML requirements and training and validations are repeated using datasets created in the training data composition phase. This phase is explained below with reference to the flow of *iterative training* shown in the top right of Figure 12.

4.2.1.3.1. Model design/adjustment step

In the model design/adjustment step, hyperparameters necessary for training (including the structure of learning models, learning algorithms and various parameters) are designed in accordance with the specifications for ML requirements and training software components are created. Hyperparameters may be adjusted in response to the results of validations and

tests.

4.2.1.3.2. Training step

Machine learning models are trained using training datasets based on designed hyperparameters.

4.2.1.3.3. Validation step

Validation datasets are input into trained machine learning models to evaluate models based on evaluation indicators (accuracy, precision, recall, F values, etc.) of learning models. Generally, several machine learning models are trained with several hyperparameters and evaluated relatively to judge their adequacy.

4.2.1.3.4. Postprocessing step

In the postprocessing step, models are transformed to adapt trained machine learning models to in-operation environment (inference servers, edge devices, etc.) and learning models for implementation are generated.

The following are some specific cases.

- Transformation of models appropriate for in-operation environment (computational performance)
  - Changes of calculation precision
  - Compression and speed-up of models (distillation, quantization, etc.)
- Optimization of models appropriate for in-operation environment (efficient inference)
  - Compiling of models (parallelization, vectorization, etc.).

The figure of a process model envisioned in the Guideline expects this postprocessing to be placed immediately after the iterative training phase and the quality check/assurance phase to be carried out under conditions close to the time of in-operation. However, postprocessing is often carried out as a part of the process of building the whole system after the quality check/assurance phase of machine learning components in actual development. In this case, the quality checked in the quality check/assurance phase may change (or deteriorate) at the time of in-operation, and it may become necessary to re-validate numerical equivalence in the test stage of the whole system.

4.2.1.4. Quality check/assurance phase

In the quality check/assurance phase, tests are designed in accordance with the specifications for ML requirements to carry out evaluations (tests) of learning models for implementation. This phase is explained below with reference to the flow of *quality*

check/assurance shown in the bottom right of Figure 12.

4.2.1.4.1. Test design step

In the test design step, programs required for tests are created and test data is added in accordance with the specifications for ML requirements. Test data to be added may include cases where data is generated by means of pre-processing data augmentation and cases where trained learning models generates data which tend to make erroneous inferences.

4.2.1.4.2. Test step

In this step, the accuracy of learning models for implementation is evaluated based on regular indicators (accuracy, precision, recall, F values, etc.) using test datasets, and its stability is evaluated using other data such as data made by adding noises to ones in the datasets. In cases where any evaluation result does not satisfy the specifications for ML requirements, the process returns to the previous phase to adjust learning models and training datasets and modify the specifications for ML requirements.

4.2.2. System building / integration test phase

This stage is not part of the machine learning building, but it is described here due to the order of the procedures.

In an ideal development situation, all quality requirements for machine learning components are pre-arranged through the PoC trial phase, and all achievements can be confirmed up to the machine learning building phase.

It is hoped that no problems will occur at the subsequent stage of system building and integration testing.

In actual system development, complex real environment requirements analysis may not be perfect, unexpected interactions may occur with other components, and the effects of defects in other components may spread. Quality defects in machine learning components due to these various factors often occur during the so-called integration testing stage. Moreover, especially in a large-scale and complicated system, the prior data is inevitably insufficient as test data, and in many cases, inspection in the actual environment / system after integration is indispensable. In addition, it is conceivable to reproduce the inspection for rare cases that cannot be prepared in advance in the data composition phase at the stage of integration test.

In any of these cases, if a test fails, ML requirements analyses are partially modified and the whole machine learning model building phase is repeated again. In order to avoid this enormous amount of repeated work, it is desirable to accurately record the content of quality management activities conducted in each phase of each stage of machine learning building so that the effect of modifications can be grasped correctly and partial modifications are made without fail.

4.3. Quality monitoring / operation phase

After the system is deployed in actual operation, the activity to maintain the performance by continuously monitoring the quality at the operation stage and making necessary corrections is clearly positioned as the *quality monitoring / operation phase*. Although this phase is placed outside the development phase in a more restricted sense in the waterfall V-shape development model, it already exists in the concept of system lifecycle such as the RAMS standards.

We consider that, in many cases, the quality needs to be managed during operation, from the viewpoint of 1) When qualitative requirements analysis based on prior data does not capture the actual environment and 2) When an environment at the time of operation is different from the environment when prior data was prepared, in machine learning based systems.

In various applications of machine learning, there are various patterns to update trained machine learning models in actual operation, because they cannot be classified uniformly. Therefore, the Guideline classify such patterns into two categories.

1) Pattern of carrying out re-training of machine learning in development environment and deploying it in operation environment after conducting tests.
2) Pattern of carrying out additional learning automatically in operation environment so that the system is self-adapted.

Section 6.9 summarizes the principles for these patterns individually.

5. How to apply this guideline

5.1. Basic application process

In this section, we overview the process of applying the Guideline.

Although this section focuses on the build-out of machine learning components, it includes processes such as analysis of whole systems which is deemed to have been carried out from the past in the scope necessary for explanation. Therefore, in cases where a specific development process has been built for the adaption to international standards and a person in charge of development can explain its adequacy, modifications (addition of any necessary stage listed in this section based on that existing process) may be made.

5.1.1. Identification of functions in charge in machine learning component system

This paragraph can be skipped in cases where the functions which machine learning components should fulfill within the system such as safety are fully identified in conventional system development processes (e.g. IEC 61508 [12]).

In cases where qualities in use of overall machine learning systems have been unidentified, qualities in use and external qualities of machine learning components are identified through the following process.

5.1.1.1. Prior examination on safety functions/check of applicable standards

- Whether a machine learning based system requires safety functions over a certain level (roughly higher than SIL 1/non-negligible personal injury is envisioned) is examined in advance.
- In cases where safety functions are required, applicable functions should be selected from IEC 61508 or standards of each application field and follow that process.

5.1.1.2. Identification of system function requirements (purpose/goals)

- The purpose of using the system, the scope of envisioned use environment and KPI to be achieved are identified at an overview level.

5.1.1.3. Examination on risk scenarios related to system use

- Examine a possibility that the system's functional requirements are not achieved or

the system becomes incompliant to the purpose of use or social requests and list damages that may be caused in that case and other disadvantages. Respond to risk assessments.

5.1.1.4. Examination of requirements for characteristics of qualities in use of overall system

- Examine the quality required when the system is in use using quality metrics of any system.
  - As an example of means when there is no appropriate option, the three axes of external quality characteristics listed in Chapter 3 shall be applied to qualities in use to judge which disadvantage examined in the previous section is applicable. Then, the degree of severity thereof is judged based on the standards mentioned in each section of Chapter 3 to correspond them to the quality levels.
  - The maximum level should be identified for each of the three axes, and they shall be the level of qualities in use which the system should achieve.
- However, in cases where existing safety standards are adopted in Section 5.1.1.1, a risk avoidance level in relation to physical damages shall be decided based on quality indicators (functional safety, etc.) applicable to the existing standards.

5.1.1.5. Identification of components that contribute to the achievement of system component design / functional requirements and quality characteristics during use

- A combination of system components is designed to differentiate functions achieved by machine components and conventional software from those achieved by machine learning components.
- Moreover, it is analyzed which component of system the achievement of the characteristics of qualities in use depends on.

5.1.2. Identification of required level for achieving external qualities of machine learning components

- A level of achievement of external qualities is identified by means of the following procedures. At this time, machine learning components should specifically contribute to qualities in use.
- As regards risk avoidance, when any conventional standard for functional safety is applied to risks of physical or human damages, an analysis based on the conventional standard has priority, and a functional safety level to be achieved by machine

learning components as software based thereon shall be replaced with a level of achievement required for risk avoidance of machine learning components.

– Other cases are defined as follows:
  - Basically, a characteristic level of qualities in use required for the whole system shall be a level of achievement of external qualities required for machine learning components.
  - In cases where undesirable output from machine learning components is monitored and corrected (modification by overwriting the output) by means of software components (Figure 9) processed serially or parallelly with machine learning components and sufficient quality is judged to be ensured for said software components by means of any conventional method, a quality characteristic level required for machine learning components is one level lower than the whole system level.
  - When machine learning components do not contribute to or interfere with the achievement of the quality characteristic level required for the whole system at all, no level of achievement of said quality characteristic should be set for machine learning components (Level 0).

5.1.3. Identification of level required for internal qualities of machine learning components

– For each item of the internal quality characteristics listed in sections of Chapter 6, a level required for each characteristic is deducted from the level of achievement required for the determined external quality characteristics described in the preceding paragraph in accordance with the relationship set in Chapter 6.

5.1.4. Realization of internal qualities of machine learning components

– A method of realizing the level required for the internal quality characteristics mentioned in the preceding paragraph is examined in accordance with each sections of Chapter 7.
– The realization of each characteristic is assured by means of the technologies and processes listed in each paragraph and other technologies deemed to be equivalent thereto.

5.2. (Informative) Entrusting AI developments

The content of this section is informative.
The work explained in the previous section sometimes cannot be covered within one

organization, thus requiring a development entrust. This section mentions points development entrusters and development entrustees should pay attention to when they work together, especially in dealing with AI specific aspects. Regarding the contract issues around the ownership of IP and sharing of compensation for damage, please see Section 10.1.1 “Contract Guidelines on AI” by Ministry of Economy, Trade and Industry.

5.2.1. Exploratory approach

When a development entrust starts for some machine learning component, it is often difficult for entruster to clearly provide KPI and acceptance criteria to entrustee. Therefore, an exploratory approach is required. The difficulties of this approach is similar to the problems we encounter when applying so-called agile development to the product development process that requires strict cost management and quality. Thus, “enterprise agile practices” are expected to be beneficial.

For example, the PoC phase described in Section 4 can be viewed as *Inception* phase as described in DA (Disciplined Agile[7]). In this phase, the development entruster and the development entrustee should build a consensus on the purpose of the PoC phase of AI (PoC exit criteria) and deliverables to the next phase. The deliverable list (e.g., decision on test strategy and basic architecture) recommended in *Inception* phase in DA can be utilized for this activity. It is highly important to set goals for the PoC stage (stage gate). To prevent any missing out important points, and to appropriately promote subsequent phases, the following two aspects should be covered in the PoC phase. i.e., *requirement clarification* and *feasibility study* (see Figure 14). For both activities, two perspectives should be taken care: *Terminate condition of PoC* and *PoC outcome to be provided to the entruster*.

[7] A summary of best practices for businesses to adopt agile development. It was acquired by PMI (Project Management Institute) in 2019. https://disciplinedagiledelivery.com/

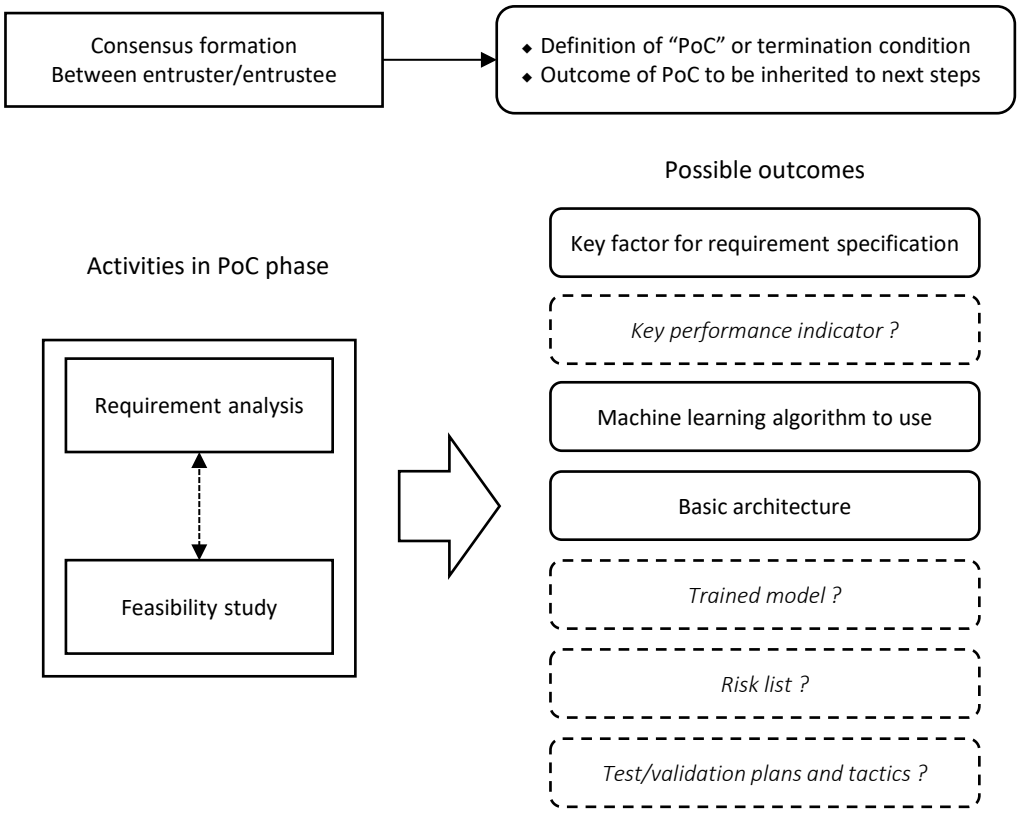

Figure 14: Activities and deliverables in the PoC phase

5.2.2. Role clarification of entrusters and entrustees

The clarification of roles to be performed by development entrusters and development entrustees, for each work in the process described in Section 5.1, will protect the both side from risks and prevent overlook of necessary works, leading to a quality assurance of machine learning components.

One example of developing tailor-made AI from scratch is shown below. Terms XXX, YYY, etc. are to be filled in appropriately.

Table 3: Example division of roles

Step	**Task for development entruster**	**Task for development entrustee**
1. safety standards	To examine primarily checking applicability of existing standards	To confirm the examined result
2. Functional requirements	To describe/provide in	To review provided document

of the system	natural language document, based on the requirements of XXX. Qualitative expressions of qualities in use to be included.	to clarify the purpose and goals.
3. Risk scenario	To prepare risk list for some similar products YYY Presentation of constraints	To analyze based on some established method of ZZZ, Update where necessary.
4. Qualities in use / external qualities	To present priorities and constraints between quality aspects (e.g., quality characteristics indispensable for the given product)	To assert quality levels to be targeted. If applicable, use three external quality characteristics.
5. Design of system components	To review and confirm	To perform system design. For reviews by the entruster, provide materials AAA.
6. Levels for external quality aspects	To review and confirm	To identify and propose target levels
7. Internal qualities	To review and confirm	To examine and propose the levels of target quality aspects and methods for achieving these.
8. Realization of internal qualities	To provide dataset sources. To review test reports	To perform quality enhance methods examined above and report.

The details in the table would differ depending on the project, and the contents once decided may change, when the work is repeated.

(Example 1)

In some cases, no specific KPI nor quality target can be presented by entruster, and *using particular training dataset* may be the only clear request as a practical measure (the condition defined in the contract). In this case, additional training data is likely to be required as a result of conducting an analysis on internal quality related to data described in Section 6. It is desirable for the both sides to recognize such a risk and reach an agreement on necessary arrangements (their roles) in advance, considering Point 7 in the above table.

(Example 2)

When the development entruster *lacks understanding* or *has ambiguity* about the quality to be achieved by the machine learning components in the early stages, such activities as *Presentation of priority and constraints* in the above table can be hardly made. In this case, the development entrustee side may present an initial idea as to qualities in use (in a bottom-up manner taking the outline of system design into consideration) in the first PoC phase. In the subsequent main development phase, the roles are to be updated as the entruster vision becomes clearer.

In the case of the PoC phase, the content of each step in Table 3 may change, but entire step is not unnecessary. On the other hand, in the main development phase, the loop within Step 8 is usually sufficient, if the goals for the stage gate (PoC exit criteria) have been set and cleared appropriately.

However, depending on the projects, the goals for the PoC stage gate must be dropped or compromised. This means that there may be no clear separation between the PoC phase and the main development phase, leading to uncertainty of the outlook for quality management. In this case, the development entruster shall take the risk, and it is usually left to the development entruster's judgment, whether to accept it and what to do with the work division for mitigating the risk.

5.2.3. Notes on determining the detailed roles

We explain here the results of extracting and examining the items to be noted from the perspective of AI with reference to "Guide to Quality Assurance in Connected World" [130] of IPA/SEC, to concretize works and make arrangements in the development lifecycle as described in the previous section.

1) System and organization that can fulfill accountability

In AI development, quality evidence are often based on the process perspective only. It is so important to clarify the definition including *who can approve it*. Also, we should consider underlying risks in the process.

Example: Regarding the training dataset preparation, it is not enough to describe the preparation of raw data provided by entruster. We should care about who will perform preprocessing (e.g., data cleansing) and who will (and how) confirm that those preparation has been appropriately achieved.

It is important to consider and specify *compromises and logics the both sides can agree on* in advance in the face of time and cost constraints.

2) Test

Even if the training of models is completely up to the entrustee side, the entruster usually must have deep understanding of the content in the following phase; i.e., *quality check/assurance*. For that purpose, it is desirable to not only define evidence but also make arrangements as to the method and scope of tests, and what to give up in terms of effectiveness and efficiency, so that the both sides feel convincing as much as possible when the contract made.

It is important that both sides cooperate for the purpose of *final quality improvement*. For example, any act just focusing on clearance of verification, (such as using test datasets presented by entruster for training), must be strictly avoided.

3) Quality management during operation

Regardless of immediacy of model update, continuous quality management is indispensable for the machine learning component even after its release, as described in 4.3 “Quality monitoring/operation phase”. Therefore, it is necessary to include the design and implementation of optimal quality management methods during operation in accordance with system functional requirements in the scope of works.

For example, both sides may discuss on identification of *data to be obtained for performance evaluation* and *environmental data which cannot be grasped completely during development*, and the system implementation for obtaining and saving those data during operation.

5.3. (informative) Notes on delta development

Existing software components are often recycled not only in machine learning based systems but also in systems and services which use software components. In machine learning, additional learning is applied to certain data to customize it based on trained models. It is complicated to guarantee the quality of recycled products.

As a fundamental principle, in order to ensure both conventional functional safety and the quality of machine learning covered in the Guideline, the guidelines for assuring quality in situations where a new system is used (context) is consistent from the analysis of conditions in use through the definition and implementation of functional requirements to tests with regard to the quality of systems which use recycled software components. There are, for example, the following methods to realize this consistency, any of which should be chosen in accordance with a quality level required by the system and the state of components provided.

1. A new quality management process is established in the context of new system including detailed checks of datasets in which recycled machine learning components (older datasets) are installed, and quality management is carried out independently from older components.
2. Quality management activities equivalent to those described in the Guideline are carried

out with respect to recycled machine learning components. When a level of quality management higher than the requirement of the new system is carried out, especially when we can clearly confirm that an envisioned condition in use is a subset of envisioned situations of older components (including the case where they are the same) with regard to *sufficiency of problem domain analysis*, it is necessary to check if the quality is not deteriorating due to additional learning as needed with reference to records of original quality management.

It is difficult to apply this method if said components are not conscious of quality from the beginning, since quality management of older components should have been carried out and recorded sufficiently. Moreover, because the analysis of conditions in use implicitly assumes certain context of use, it may be difficult to judge comprehensiveness.

Furthermore, a developer that provides machine learning components as general-purpose parts can improve reusability by completing quality management activities in advance in anticipation of this type of recycling and by clarifying the envisioned quality levels.

3. When a relatively low-quality level is required for application, quality management activities with a focus on the test phase should be examined on the assumption that existing datasets are considered as an unknown black box. For example,
   - For 6.1 “Sufficiency of requirement analysis” and 6.2 “Coverage for distinguished problem cases”, make a fresh analysis for a new system to set a new quality targets.
   - For 6.3 “Coverage of datasets” and 6.4 “Uniformity of datasets”, prepare new test datasets based on new result of requirement analysis, and check achievements of these characteristics to a certain degree during the test phase.
   - For 6.6 “Correctness of trained model” and 6.7 “Stability of trained model”, If additional learning is carried out, make an evaluation in the validation/test phase using performance indicators obtained during additional learning, and using training datasets added based on new requirement analysis. When additional learning is not carried out, make a validation based on new result of requirement analysis in the test phase, or to perform evaluation during integration tests on the whole system.

   However, the current descriptions of Chapter 6 and Chapter 7 reveal that it is difficult to carry out sufficient quality management activities by using only sampling-type tests of training results without analyzing datasets used for training. It is practical at this stage to obtain detailed information on older datasets and combine it with a white box approach described in the previous paragraph.

6. Requirements for quality assurance

This chapter sets the following nine internal qualities as the quality management characteristics to manage quality of machine learning components mainly for the purpose of the achievement of the two qualities in use, *safety* and *AI performance*. Section 11.1 explains the analyses leading to the setting of these internal qualities. We will further analyze the other quality in use *fairness* and show additional internal quality characteristics specific to the quality in Section 8.

6.1. A-1: Sufficiency of problem domain analysis

6.1.1. General

Sufficiency of problem domain analysis means that usages of a target machine learning based system are analyzed sufficiently and every requirement for the system is captured (described in Section 1.7.1).

Problem domain analysis is important especially in the early stage of development of a conventional software which is used for a safety application. The main purpose of requirements analysis for the development of machine learning based systems in this guideline are as follows.

1. Sufficient identification of cases in which risk management is needed, mainly in applications required *safety*.
2. Sufficient identification of attributes of objects for which inequality is not allowed, mainly in applications required *Fairness*.
3. Sufficient analysis of real world in order to verify that training datasets and test datasets are comprehensive and appropriate extractions of the real world, to all requirements including *AI performance*.

Sufficiency of requirement analysis is always required not only for machine learning based systems but also equipment and services controlled by software.

However, if any situation where machine learning based systems are used is overlooked at this stage, there are few opportunities for taking note of an error from the data collection stage to the actual training/test stages, so that it is likely to cause malfunction that occurs for the first time in the stage of final system test in real environment or the actual operation stage.

On the other hand, if the details of all possible situations where machine learning based systems are used are analyzed including minute differences, the analysis results can be implemented as a regular software component, and there is no merit in using a machine learning technology.

To put it another way, it is impossible to make comprehensive and thorough analyses in advance for such applied systems. That is why there is demand for having systems acquire knowledge through machine learning in any way.

From these two viewpoints, it is extremely important to appropriately set *levels of detail of requirements analyses* in machine learning based systems in terms of both quality assurance and feasibility and efficiency of implementation.

They correspond to the judgments of *what do humans need to analyze* and *what should we have machine learning acquire*. The maintenance of appropriate balance between these two viewpoints is an important goal of requirements analysis in quality management of machine learning.

6.1.2. Approaches

The two goals of this internal quality aspects described in this paragraph are;

- Clarifying what is required for machine learning components; and
- Clarifying the limited scope which machine learning components have to deal with.

6.1.2.1. Extracting and listing attributes (characteristic viewpoints) and attribute values (specific features)

At first, inputs in real world which can be extracted as specific characteristics are organized from the aspect listed below. Each identified viewpoint will be referred to as *attribute*, while the specific features belonging to the viewpoint as *an attribute value*.

The following examples can be given as attributes and their viewpoints.

1. Attributes that characterize the differences in output required for machine learning components as functional requirements.
   - In supervised machine learning, *supervision labels* are different.
   - For example, in the recognition of characters (numbers), 10 distinctions from 0 to 9. In anomaly detection, whether there is any abnormality is detected.
2. Attributes whose output is the same but require machine learning components to explicitly respond at a functional specification level.
   - One example is a case where, in character recognition, the specifications are set that both l *and* ℓ and l *and* / should be recognized correctly.
   - Another example is that, when the specification says that various kinds of obstructions such as *pedestrians*, *bicycles* and *automobiles with crossing movements* are to be avoided in automated driving, each kind of object should be identified as a separate attribute value. Or, *gender* and *age groups* in the scoring for recruitment that

requires fairness apply to this paragraph.

3. Attributes for which deterioration risks of performance expected for machine learning components are likely to be different significantly in real operation
   - Some examples include *climate*, *distinction of day and night*, *backlight and front light* and *movement speed* in the case of outdoor object recognition.
4. Elements whose expected output is the same due to the characteristics of machine learning, but it is difficult to find a common ground by the model after learning or they can be recognized as different internally.
   - For example, in the case of outdoor object recognition, different characteristics of objects may be captured in the daytime and nighttime, or another example is the distinction of the horizontal and vertical installation of traffic lights.
   - Another example is that different forms of handwritten numbers such as 1 *and* / are categorized into several characteristics which differ greatly depending on countries of origin. In order to recognize all different types, it is necessary to intentionally include them at least in training datasets and test datasets as different general populations, even if they are not clarified in the functional specifications.
5. Characteristics which make it difficult to specify the method of evenly collecting learning data as a process when data is collected.
6. Differences which humans can easily capture, though there are no differences as described above, e.g. Kinds of animals, skin color, etc.
7. Object which humans cannot recognize or explain their characteristics with word, but it is possible to extract a sufficient number of samples without bias. For example, it is difficult to analyze all possible ways of describing the number 8 in advance. However, if people do not intentionally write various different styles of the number 8 and we have a sufficient number of randomly extracted samples, they are expected to be unbiased samples.
8. Cases where it is extremely easy to mechanically cover and complement differences. For example, once required specifications are established, it is possible to mechanically generate samples of parallel shift of images, color changes and expansion of rotations within a certain range, and intentionally synthesize training data and test data.

We make up a list for candidates of possible attributes from requirements analyzed on the overall system, and then we should determine whether each attribute is picked up for explicit target of management by the developers or is *left to machine learning* without picked up. To determine that, the level of risks related to values of each attribute and the type of the application using machine learning (results of PoC trials may be useful as a reference) should be considered. In general, the characteristics shown earlier in the above list (eg 1～3) are highly likely to be extracted as attributes. In contrast, it is often reasonable to leave the characteristics like listed lower (in 7～8) for machine learning without extracting them as

attributes. However, the diversity of data types corresponding to attributes left to machine learning will be examined in the third internal quality *Coverage of datasets*, so that it is necessary to take this point into consideration in the analysis, too.

6.1.2.2. Combinations of attributes that should be excluded

Impossible combinations of attribute values should also be examined. The purpose of examination is to distinguish cases which should be excluded as functional requirements such as *snow in summer (in Tokyo)* from cases which should be handled as functional requirements such as *snow coverage in winter* which occurs very rarely. It is important to make this distinction, because it will become impossible to determine whether lack of data is acceptable and to explain *the quality of quality management method itself* in subsequent quality management.

To be more specific, after attributes and attribute values corresponding thereto are listed, impossible combinations of attribute values are selected based on system requirements.

6.1.3. Requirements for quality levels

For *sufficiency of requirement analysis*, it is required to handle the following three levels in accordance with the level of external quality.

The correspondence between external qualities and internal qualities regarding the required level is as follows.

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: More than Lv 3 (some requirements will be added to Lv 3)
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each level of internal qualities are as follows.

- Lv 1
  - Examine and record the major cause of possible deterioration of quality.
  - Based on the examination results, design data and reflect it in necessary

attributes.

– Lv 2
  - Analyze risks of deterioration of quality in use in overall system and their impact with a certain level of engineering coverage and record the results in documents.
  - Analyze if any measure is required for each of those risks, and analyze attributes related to the risk which are contained in an input to machine learning components.
  - Analyze and record the application-specific characteristics of environments which will generate machine learning input, with regards to the difficulty for machine learning and other aspects.
  - Examine sets of attributes and attribute values, based on the results of those analysis and record the background of such decisions.
– Lv 3
  - The following activities are carried out in addition to those listed in Lv 2.
  - Investigate documents on own past examination results and those of others with regard to elements to be extracted as characteristics of system environment and record the background of examinations leading to the extraction of necessary subsets.
  - Investigate past examination results in line with application fields of systems with regard to deterioration risks of qualities in use of overall systems and record the examination results including the background of selection.
  - Moreover, extract deterioration risks of qualities in use of overall systems using engineering analysis such as Fault Tree Analysis and record their results.

6.2. A-2: Coverage for distinguished problem cases

6.2.1. General

On the basis of the sufficiency of requirements analysis described in the previous paragraph, it requires careful consideration of data design as *Coverage for distinguished problem cases* in order to secure sufficient training data and test data with respect to various situations systems need to respond to. More specifically, the number and details of combinations of attribute values focused in the stage from training data preparation to testing process is considered at this stage.

In an ideal situation, for example, if there are sufficient data corresponding to all combinations of attribute values (direct products of attributes) for combinations of attributes that are supposed sufficient as described in the previous paragraph, it can be said that it covers all possible situations in the real world. However, the number of attributes may reach 10 or more in actual system development so that the number of combinations of attribute values

may often reach thousands or millions. In this case, it is crucial to consider appropriate *coverage* and *design* for quality management in this paragraph.

It is important to focus on two viewpoints in actual quality management. First, combinations of attributes that may cause malfunction or misjudgment need to be reliability addressed at the training or testing stage. Second, at the same time, it is necessary to cover as much as possible all situations which the machine learning based system to be implemented may encounter during operation from the viewpoint of both the quality of training and the quality of deliverables.

Such a problem has been addressed as a task of *test design* in developing conventional software. However, in machine learning where not only the testing process but also the implementation process is performed based on datasets, it is unique that the same task is required in the implementation process.

Some practical solutions to the excessive number of combinations for test design have been already presented in conventional software engineering. This guideline aims to provide practical and sufficient training and quality tests by applying such existing knowledge to machine learning applications.

6.2.2. Approaches

Firstly, if the number of attributes to be recognized is very small and the total number of all attributes (the impossible cases described in the previous paragraph are deducted) is only 10–20, their combinations are named as *cases*, and they are checked if all cases are included in test datasets and training datasets in later stages.

On the other hand, if there are too many combinations when the total of those attributes are used or sufficient data cannot be acquired especially in rare cases, combinations of some attribute values should be extracted under certain standards and set as *cases* to cover all those combinations. For example, in the example of traffic lights listed in Example 1 in Section 1.7.1, such combinations as *daytime green*, *daytime yellow*, *nighttime red*, *daytime rainfall*, *nighttime rainfall* and *red signal in rain* are extracted and called as *cases* to make sure that data applicable to these combinations is included in test datasets. Even if all attributes cannot be covered completely, it is possible to intend to achieve a certain degree of *coverage* by avoiding cases where high-risk situations such as *nighttime rainfall* are not included at all in datasets or eliminating cases where *red signal* is not included in training at all. When considering such *simplified coverage*, the data of *red signal and daytime rainfall* are included in the several cases such as *daytime rainfall*, *daytime red signal* and *red signal and rainfall* at the same time.

Furthermore, completeness should be guaranteed more specifically in applications that require high quality by introducing mathematical *coverage criteria* to such extraction works. In the case of black box tests in software engineering, methods such as the sampling rate of

random sampling, the orthogonal table and pair-wise test and the *coverage criteria* based on them are known, we should select appropriate means for each application.

6.2.3. Requirements for quality levels

The handling of the following three levels is required for coverage of distinguished problem cases in accordance with a level of each external quality.

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 should be examined.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv 1
  - Set cases for each of attributes corresponding to major risk factors.
  - Moreover, set cases corresponding to combinations of composite risk factors.
  - Furthermore, extract attributes of differences in particularly-important environmental factors and prepare cases corresponding to combinations with serious risk factors.
- Lv 2
  - Satisfy all requirements listed in Lv 1.
  - Particularly-important risk factors should satisfy, in principle, the standards for pair-wise coverage. To be more specific, a case of combining *an attribute value of combination of those factors* and *individual attribute values included in all attributes other than those to which the attribute value belongs* should be included.
- Lv 3
  - Based on engineering consideration, set standards for coverage of attributes and establish sets of combinations of attribute values that satisfied standards for coverage as cases.
  - The level of strictness of the standards for coverage (pair-wise coverage, triple-

wise coverage, etc.) should be set taking into account system usage and risk severity. Standards can be set individually for each risk where necessary.

6.3. B-1: Coverage of datasets

6.3.1. General

The term *coverage of datasets* means that a sufficient amount of data is given to each cases covered by establishing the standards as described in the previous paragraph without omissions for the input possibilities corresponding to each cases.

When conventional software is developed, the details of all characteristics in real world which software operations depend on are captured at any stage from the machine learning requirements analysis phase to the implementation phase and reflected in software components in the form of conditional branching or calculation formula. On the other hand, when a machine learning component is built, differences in details exceeding a certain level are not captured as attributes of ML requirements analysis or *labels* at the time of training but are reflected in ultimate operations through the training phase of machine learning as a distribution of datasets used for learning, as described in Section 6.1. The purpose of establishing this characteristic axis is to guarantee that no inappropriate learning behavior occurs due to lack of data with regard to characteristics of those unidentified details as *attribute values*.

6.3.2. Approaches

The main purposes of this quality aspects are to achieve sufficient level of *quantity* and *coverage over input situation*. However, since there are characteristics whose minor differences have not been identified as attributes, the latter coverage often has no choice but to depend much on the process of collecting and processing data. On the other hand, when the standards for coverage are introduced in Section 6.2, it would be possible to test if attributes which *are not included in cases* are distributed without bias.

Moreover, as regard the former quantity, it is important to appropriately define granularity of how to set the cases described in the previous section, but it is possible that sufficient data for specific cases cannot be obtained, for example, in rare cases where the frequency of occurrence is low. In those cases, it may become necessary to take measures in the whole development process for evaluating the response status by means of tests on the whole system by partially abandoning *coverage of datasets* for cases that lack specific data and after covering looser coverage standard.

6.3.3. Requirements for quality levels

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2～4: Requirements to be added to Lv 3 should be examined.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 2 or above
  - AIFL 2: Lv 3
    (note: We consider that this quality characteristic is important in dealing with data bias, which is an important factor that impairs fairness.)

The requirements for each required level for the above internal quality are as follows.

- Lv 1
  - Consider the source and method of acquiring test datasets to ensure that no bias is found in application situations.
  - Extract samples without bias from original data for each case to ensure that no bias is found.
  - Record activities carried out to prevent bias from entering.
  - Check that there are sufficient training data and test data for each analyzed case in the training phase, validation phase, and so on.
  - When sufficient training data cannot be acquired for any case, review and loose the coverage standards and record what should be checked individually by system integration tests in line with the original standards.
- Lv 2
  - The following activities are carried out in addition to those listed in Lv 1.
  - Grasp an approximate probability of occurrence for each attribute value or each case.
  - Check if acquired data is not deviated from the distribution.
  - Positive check other than acquisition methods should be made regarding the coverage of the data included in each case.
    - For example, in each case, when there is any attribute not included in that case, extract the distribution related to attribute and check if there is no

significant bias.

– Lv 3
  - ➢ Acquire certain indicators for coverage of data included in each case in addition to those listed in Lv 2.
    - ✧ For example, check if there is no correlation between data other than attribute values included in combinations of cases using feature extraction or any other technique.
    - ✧ Or consider an expected distribution of attributes not included in each case, and analyze and record differences.

6.4. B-2: Uniformity of datasets

6.4.1. General

Evaluating only the *coverage of each case* in the previous paragraph does not always mean that all datasets are good sampling of the overall environment expressed by input data. When the probability of occurrence differs significantly from case to case, simply preparing samples for each case will generate datasets with a large bias as a whole, which may significantly impair performance, especially in terms of AI performance. On the other hand, when performance is required for rare cases that probability of occurrence is very low, it cannot be generally coexist the preparation of the practical amount of uniform data without bias for all input and the preparation of the sufficient amount of data for rare cases. For example, when 100 training data are required for an event that occurs at a frequency of one millionth, it is not usually acceptable that the number of cases of all unbiased data is 100 million. From this viewpoint, it is considered that the uniformity in this paragraph needs to make an appropriate compromise in some cases, contrary to the coverage in the previous paragraph.

In general, it is required to prepare sufficient training data for combinations of attribute values with risks which should be avoided by making correct judgments when risk avoidance is strongly sought. When such a risk occurs on rare occasions, an enormous amount of data might be required for training all other cases with a sufficient *amount of data*. In such case, it is quite conceivable to *focus* on training of particularly rare risky cases.

On the other hand, when overall performance (AI performance) is required, by focusing on training rare cases more than the actual probability of occurrence, the inference accuracy deteriorates in other cases, and it may also worsen average performance. In such case, it is not always appropriate to deeply explore the coverage of detailed cases in the previous section.

Moreover, when fairness is strongly required, it may change whether the training should be artificially equivalent between cases or should be randomly trained according to the distribution of extracted training datasets, depending on a type of fairness required.

6.4.2. Approaches

The basic concept itself is to focus on cases of *overall* in the topic of coverage in Section 6.3.2. While taking care not to bias in the process of acquiring whole datasets, it is necessary to monitor the frequency of occurrence of each attribute value as appropriate.

Rather, as stated in general, in this section, it is important to consider how to coexist with the coverage in the previous section and data design.

6.4.3. Requirements for quality levels

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- *Safety*
  - AISL 0.1: Lv S1 or above
  - AISL 0.2/1: Lv S2 or above
  - AISL 2∼4: Requirements to be added to Lv 2 will be examined.
- *AI performance*
  - AIPL 1: Lv E1 or above
  - AIPL 2: Lv E2 or above
- *Fairness*
  - AIFL 1/2: Lv E2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv E1
  - Same as Lv 1 in *Coverage of datasets* in the previous section.
- Lv E2
  - Same as Lv 2 in *Coverage of datasets* in the previous section. However, assumed probabilities of occurrence are compared with the whole sets of assumed events.
- Lv S1
  - Regarding the amount of data for each case considered in L1 of the previous section, explicitly check if there is a sufficient amount of data for risk cases.
  - When data of rare cases is insufficient for training, comparing the amount of the whole sets of training data with a probability of occurrence of rare cases, consider focusing on learning of rare cases. However, especially when Lv E2 is required, with prioritized, the impact of reduced training of other cases on whole system quality should be considered.
- Lv S2
  - In addition to what is listed in Lv S1, estimate and design in advance the amount

of data of each case, based on the estimated probability of occurrence for each risk event / case.

6.5. B-3: Adequacy of data

6.5.1. General

Adequacy of data means that the data used in the machine learning training and testing process is free of errors and inappropriate data, and includes a variety of internal perspectives.

The data quality defined by SQuaRE (ISO/IEC 25012) [8] can be classified in terms of data-specific and system-dependent aspects used as follows. Of these, the underlined terms are considered to be particularly important as the source data for building machine learning.

- Data-specific quality characteristics
  - ✧ Accuracy (1)
  - ✧ Completeness (2)
  - ✧ Consistency (3)
  - ✧ Credibility (4)
  - ✧ Currentness (5)
- Data quality characteristics from both data-specific and system-dependent perspectives
  - ✧ Accessibility
  - ✧ Compliance
  - ✧ Confidentiality
  - ✧ Efficiency
  - ✧ Precision (6)
  - ✧ Traceability (7)
  - ✧ Understandability
- Data quality characteristics from the system-dependent perspectives
  - ✧ Availability
  - ✧ Portability
  - ✧ Recoverability

Also, in terms of the machine learning process, there are two distinct factors:

A) Data Selection Appropriateness:

Existence of each data in the dataset is appropriate for the policy fixed by the coverage and uniformity of the dataset (B-1, B-2). For example, each data does not contain measurement errors or is not outliers that should be removed.

[(1) Accuracy, (2) Completeness, (4) Credibility, (5) Currentness, (6) Accuracy, (7) Traceability]

B) Appropriateness of labeling:

For each data in the data set, the information added in the data set preparation stage is appropriate.
[(2) Completeness, (3) Consistency, (4) Credibility]

These aspects are generally considered to correspond to the properties of SQuaRE as shown in square brackets.

Authenticity, which is often pointed out as data quality in the field of artificial intelligence, and is often important from the perspective of security, can be considered to correspond to *Credibility* (4) and *Traceability* (7) in the above quality characteristics.

6.5.2. Approaches

Although the validity of the data is ultimately aimed at *obtaining good machine learning elements*, from the perspective of quality management, it is basically considered to be evaluated against the requirements for the data expected by the system, which are defined by the internal quality characteristics from A-1 to B-2.

In the specific evaluation of this quality characteristic, it is necessary to evaluate various viewpoints in the inspection of the data itself as well as in the management of the construction process.

6.5.2.1. Unification and Scrutiny of Labeling Policies

Although the labels to be assigned as teacher data themselves are specified in A-1 Completeness of Problem Domain Analysis, there can be various fluctuations, confusions, and ambiguities in the actual data. For example, in image feature detection, the size and distance of objects to be extracted as labels, and the handling of blocking conditions of overlapping objects need to be clarified in terms of functional requirements. If these points of view are not consistent among workers, fluctuations in the labels may lead to reduced accuracy in the training process and inaccurate inspections in the testing process.

In addition, when the requirements are extended due to repeated trials in the PoC phase or rework due to insufficient accuracy, the labeling itself needs to be extended or modified. However, relabeling work is generally expensive, and even here, there may be inconsistencies in labeling among workers. Furthermore, when acquiring additional data, the environmental

conditions of the data itself may not be consistent.

From these perspectives, it is important to conduct a thorough study as early as possible in the PoC phase, to solidify the labeling policy in as much detail as possible, and to record it in writing, in order to ensure traceability of quality control.

6.5.2.2. Consistency Checking and Rechecking of Data Sets

In the construction of a machine learning system, it may be possible to use existing pre-measured data sets or labeled data sets. However, since the validity of data is determined by its consistency with the requirements, the validity of existing data must be re-evaluated whenever the functional requirements or the assumption of the operating environment is changed. In addition, when data collection or labeling work is outsourced, it is necessary to conduct inspections for acceptance from the viewpoint of quality control. How such inspections are conducted needs to be thoroughly discussed in advance, even before the process is established.

6.5.2.3. Handling the Long Tail and Determining Mismeasurement and Outliers

It is possible to automatically screen for a certain amount of data being out of tendency from other data by using statistical analysis or other methods. However, whether such rare data should be taken as a meaningful training target, such as the long tail, or should be dismissed as outliers / mismeasured values may depend on the nature of the problem and the content of the individual data and may also depend on the priorities of external quality of risk aversion and AI performance. Without a clear policy or decision-making process in this regard, the data selection and labeling process will produce ambiguities.

6.5.2.4. Addressing data contamination (security and authenticity)

When training and test data can contain intentional errors or biases, or when there are malicious modifications to the measurement environment (e.g., interference with sensors), the functionality of the final system may be severely affected. Such errors are not detected in the testing process if there is contamination in the test data, and in general cannot be prevented sufficiently by system testing.

From this point of view, not only the information security protection of the data itself, but also the physical security and diversity of the data acquisition environment must be ensured from a process management perspective, and it is also important to keep records of such quality assurance activities.

In addition, at present, there is no general-purpose method to detect data contamination before training, especially in the case of procuring data from external sources. From this point of view, a system that requires a certain level of quality may have to rely on the credibility and traceability of the data set itself or the provider to prevent data contamination

6.5.2.5. Currentness of the data

In machine learning applications, performance often degrades as time passes since the acquisition of training data. Managing the currency of the training data is important to prevent such quality degradation. On the other hand, the requirement for currentness often conflicts with the amount of data available for training, and there can be a trade-off between currentness and completeness, especially when rare cases need to be handled. From this point of view, it is necessary either to consider the policy of currentness in advance, or to identify a reasonable requirement level at the PoC stage.

6.5.2.6. Processes, Organizations, and Systems

As mentioned above, in ensuring the validity of data, there are many factors that depend not only on the specific handling of individual data but also on the overall construction process and the system, including auditing. In addition, in the actual construction of a machine learning system, handling concrete data in the process of judging the validity of data often leads to more detailed requirements definition, which may have a cascading effect on the internal quality characteristic A-1 in the previous stage. In this guideline, this kind of work is classified as trial and error in the PoC stage. However, when this kind of circulatory work occurs, it is especially important to verify whether the final outcome is consistent. On the other hand, it is not practical to redo all data inspections from scratch in every trial. From this perspective, it is considered that quality control requires the creation of a system and structure to properly manage the process, including the management of changes in policies during development.

6.5.3. Requirements for each quality level

The correspondence between the external characteristic level and the requirement level of this internal characteristic is as follows.

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2～4: Requirements to be added to Lv 2 will be examined.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv 1
  - General:
    - Properly examine and confirm whether the source of the data is appropriate for the problem.
    - Organize the labeling policy.
    - Review and summarize the criteria for labeling and outlier removal in advance.
    - Determine whether the criteria are appropriate in the context of the given data, and if necessary, review and recheck the criteria.
    - When labeled data is used, the validity of existing labels should be examined in advance and confirmed by pre-testing, etc. if necessary.
  - Fluctuation of labels:
    - A consistent standard should be established among workers to judge labels, or a double check is to be performed.
  - Data contamination:
    - Consider the impact and likelihood of contamination of data sources.
  - Currentness:
    - Consider in advance whether the data set contains data from an inappropriate time period, according to the characteristics of the problem.
- Lv 2
  - In addition to Lv 1, take the following actions
  - General:
    - Incorporate the data preparation stage into the quality control process and manage it properly.
    - If data is procured externally, incorporate data preparation methods, processing methods, quality control processes, and security controls into the requirements.

- Labeling Policy:
  - ✧ Establish a control process to eliminate variations in labeling by workers.
  - ✧ Establish a process to manage label changes when data attribute definitions are changed.
- Label Fluctuations:
  - ✧ Review and document in advance the range of acceptable label variability.
  - ✧ Record labelling decisions regarding fluctuations that occur during production.
- Data Contamination:
  - ✧ Consider methods to inspect training data to the extent possible.
  - ✧ Adversarial Examples should be addressed.
  - ✧ Consider a design that detects anomalies at runtime.
    See Chapter 9 “Security” for more details.
- Currentness:
  - ✧ Integrate and manage the data preparation phase into the quality control process.
- A posteriori inspection:
  - ✧ If possible, consider applying input impact analysis, neuron firing status, and other internal information analysis, and manually eliminate obvious errors to the extent possible.

– Lv3
  - ✧ Take the following actions in addition to Lv2.
- Labeling policy:
  - ✧ Conduct and record a risk analysis of the impact of label design.
- Confirmation of label data removal:
  - ✧ Double-check at receiving inspection at the time of outsourcing, or set up and inspect the audit process in advance.
- Data Contamination:
  - ✧ Conduct and document a risk analysis of data contamination.

6.6. C-1: Correctness of trained models

6.6.1. General

The term *correctness of trained models* represents that a machine learning component functions as expected upon the input from the learning dataset (consisting of training data, test data, and validation data).

6.6.2. Approaches

The correctness of trained models is usually evaluated in terms of quantitative measures, such as accuracy, precision, recall, or F-value. These measures may overfit to the training dataset, that is, they may perform well for the training data, but badly for the data that are not included in the training dataset. Therefore, in the test phase, the correctness needs to be evaluated using a test dataset that is disjoint from the training dataset, or more generally, by applying cross validation.

Data scientists should select concrete techniques suited to the application and need to explain their selection. In Section 7.6 we will explain some examples of techniques that can be used to test the correctness of a trained model.

6.6.3. Requirements for quality levels

The relationships between these internal qualities and the required levels for external qualities are as follows:

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each quality level for the above internal quality are as follows.

- Lv 1
  - Prepare a test dataset by taking into account the coverage of data and the past experiences, e.g., obtained in the proof of concept (POC) stage.
  - Prepare a training dataset in an analogous way to the test dataset. Note that the training dataset may not necessarily follow the same distribution as the test dataset.
  - Decide and record how to deal with the incorrect behavior of a trained model

(e.g., false negative/false positive in the test) before the validation phase.
  - If a machine learning component is required to satisfy fairness, decide and record the methods and criteria to evaluate fairness before the validation phase.
- Lv 2
  - All the requirements listed in Lv 1.
  - Decide and explain methods and criteria to validate the trained model (e.g., accuracy and its threshold) before the validation phase.
  - Test the trained model using the given test dataset and additional test data (e.g., generated by data augmentation techniques).
  - If possible, analyze internal information on the trained model (e.g., the neuron coverage to evaluate the adequacy of testing).
- Lv 3
  - All the requirements listed in Lv 2.
  - Perform the validation/testing of the whole system (e.g., integration tests), especially by focusing on risky cases.

6.7. C-2: Stability of trained models

6.7.1. General

The stability of trained models indicates that the machine learning elements respond as expected to inputs that are not part of the datasets. Low stability leads to poor prediction performance for unknown inputs (poor AI performance) and increased risk due to potential serious misjudgments (poor safety). Therefore, it is important to evaluate stability, especially when safety is required. The following two issues are well known regarding stability.

- The trained model can function incorrectly when the input is far from the training data. Typically, this may be caused when the model is overfit to the training dataset.
- The trained model can behave significantly differently when a small amount of noise is added to the input to the model. Such input noise can be either random noise in nature (e.g. dirty camera lens) or adversarial perturbation caused by malicious attacks. For example, those attacks are triggered by data poisoning in the learning dataset, and by certain tricks on physical objects (e.g. attachment of tiny stickers to road signs).

6.7.2. Approaches

Stability can be evaluated/improved mainly in the following three phases in the machine learning lifecycle. Some useful techniques will be explained in Section 7.6.2.

- Training phase: A trained model's overfitting to the training dataset can be avoided e.g., by separating the validation dataset from the training dataset, by regularization techniques, by evaluating the impact of small input noise, and by monitoring the training process.
- Evaluation phase: Correctness measures (e.g., accuracy, precision, recall) can be evaluated by using synthetic data that are obtained by adding random/adversarial noise to test data.
- Operation phase: By monitoring the input to the trained model, some of adversarial input may be detected and eliminated.

6.7.3. Requirements for each quality level

It is required to record the methods applied to improve the stability of a trained model and the evaluation results of the stability. In the list below, neighboring data refers to data generated by adding small perturbation noise to original data. Examples of concrete techniques will be explained in Section 7.6.2.

The relationships between these internal qualities and the required levels for external qualities are explained as follows:

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv 1: Record the concrete techniques applied to improve the generalization performance of a trained model (e.g., cross validation and regularization are widely used to prevent overfitting to the training data).
  - For stability at Lv 1, we recommend to apply some widely-accepted techniques for avoiding overtraining and overfitting, such as cross validations and reguralizations.
- Lv 2: Record the evaluation results of stability by using neighboring data.
  - In the evaluation of stability at Lv2, we require to use some synthetic data

obtained by adding a small amount of noise to the learning datasets. In particular, it is recommended to apply techniques to prevent attacks based on adversarial examples; e.g., robustness evaluation using adversarial examples, adversarial training to train a robust model, and dynamic detection of adversarial examples. Currently, these new methods are still being studied and developed in academic research, but might be applied to system development more effectively in the future.

– Lv 3: Provide some formal guarantee to the stability for data outside datasets.
  - ➢ At Lv 3, we require to formally guarantee a certain level of stability for such data. For example, methods for certifying adversarial robustness have been studied recently and might be used in system development in the future.

6.8. D-1: Reliability of underlying software systems

6.8.1. General

The term *reliability of underlying software systems* represents that the underlying conventional software (e.g., training programs and prediction/inference programs) functions correctly. This notion includes the software quality requirements such as the correctness of algorithms, the time/memory resource constraints, and the software security. When electronic hardware such as MPU is developed, this notion also includes the hardware dependability.

In many applications of machine learning systems, open-source software is used to train machine learning models and to develop conventional software systems. Although the quality of open-source software may not be guaranteed, the developers of machine learning systems should be responsible for ensuring sufficient quality even when using open-source software.

Furthermore, the correctness of software should usually be tested under the same environment as actual operation environments. When the test environment is different from actual operation environments, it is necessary to evaluate the differences between them. In the development of machine learning systems, however, there are often significant differences between an environment used for training (e.g. computing environment with cloud and GPU) and an actual operational environment (e.g. built-in computer). Even the behaviors of numerical calculations (e.g., the accuracy of floating-point computation) can change between them. In some cases, numerical processing itself may be changed e.g. by model compression. When implementing a machine learning system, the developer needs to cope with these environmental differences that may affect the AI performance of the system.

6.8.2. Approaches

Quality management methods for conventional software systems can be applied in order

to ensure the dependability of the underlying software system.

As regard the use of open-source implementations, appropriate quality assurance measures such as those listed below should be taken to achieve the safety and reliability of the system.

Moreover, when the in-operation environment is different from the environment in the training process (e.g., when performing model compression), this Guideline recommends to test the software that reproduces the same calculations as the in-operation environment in the test phase. If this is difficult, the developer should test the whole system to check if the quality deterioration falls within the acceptable range.

6.8.3. Requirements for quality levels

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

- Lv 1
  - Select reliable software used for the machine learning system, and record the process of this selection.
  - Monitor the system's operation to check and update the selected software.
  - Examine in advance the impact of differences between the environment in the training/test phases and the environment in the actual operation phase.
- Lv 2
  - Evaluate the reliability of the software used for the system by testing.
  - If possible, use software whose reliability is SIL 1 or equivalent.
  - Prepare for the maintenance of software during its operation.
  - In the validation and test phases, conduct validation tests in an environment that simulates the environment used in the actual operation phase. Alternatively, validate the consistency of operations of the trained model between in the test phase and the actual operation phase.
- Lv 3

- Check the quality of software for SIL 1 (or a higher SIL level when required by the system).
- Perform testing or formal verification of the behaviors of the trained model in an actual environment.
- Check the consistency of those models and operations in an actual environment in any stage after integration tests.

6.9. E-1: Maintainability of qualities in operation

6.9.1. General

The term *maintainability of qualities in operation* means that internal qualities satisfied at the beginning of operation are maintained throughout operation. This concept means that internal qualities can fully respond to changes in operational environments outside the system and that any change in trained machine learning models do not cause unnecessary deterioration of quality.

A specific method of realizing the maintainability of qualities in operation depends largely on forms of operation, in particular, how to carry out additional learning and retraining. This guideline envision the following two patterns of additional learning and retraining.

(a) Cases where the results of additional learning are reflected in an operation environment for the first time after going through additional learning and quality tests outside (developmental environment) of actual systems in service (operational environment) through explicit updates of software. Solutions such as automatic driving envisioned now belong to this pattern.

(b) Cases where trained models are updated on a real-time basis during operation and updates complete without any human tests in an operational environment. Online learning used to process streaming data, language recognition and application of conversations in projects in which development and operation are integrated belong to this pattern.

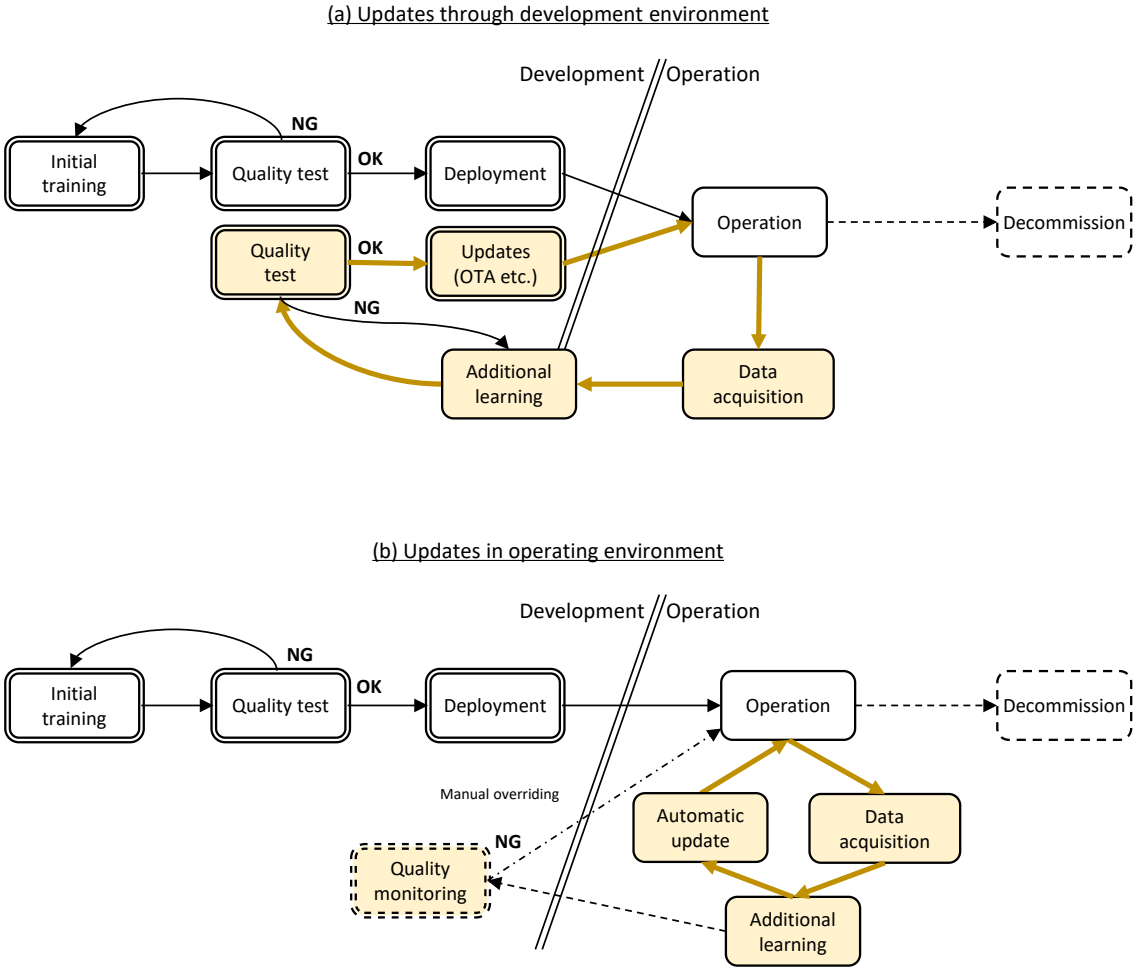

Figure 15: Forms of updates of machine learning components in operation

As described in the above figure, quality tests are always carried out before updates in the update pattern in the development environment described in (a). If the content of tests is the same as those conducted in the test phase of initial development, a certain level of quality can be maintained. On the other hand, in the update pattern of the operational environment described in (b), there is a higher risk that a model whose quality has been deteriorated is reflected in operation, because updates are fully automatic. In this case, it is important to incorporate a quality monitoring mechanism and a mechanism dealing with deterioration in an operational system.

Moreover, as regards maintainability of qualities in operation, in addition to the deterioration of overall performance, it may be necessary to give attention to a problem that a machine learning component makes a misjudgment on specific input after its update[8],

[8] In software engineering, this problem is often called “regression”. The content mentioned in this

although it used to draw out correct answers prior to the update. We might think that it is enough if the external qualities described in Section 1.5 improve or are maintained as a whole, but in actual industrial fields, it may be difficult to accept that any component which was implemented and used to operate correctly stops operating properly at a later stage. On the other hand, additional learning inevitably draws out different output values from past input due to the characteristics of machine learning based systems. Therefore, it is necessary to examine in advance how to handle additional learning in the form of operational guidelines.

Moreover, though the Guideline do not cover directly, it is required to give consideration to legal positions of contracts and privacy of data in real environment used for additional learning, when operation is examined. From the viewpoint of quality management, it is desirable to save all data used for training including additional learning and use it for validating and monitoring the quality and verifying performance deterioration when a software component used to give correct answers. However, some restrictions on handling of data may be imposed from the viewpoint of privacy in actual applications. When data which is prerequisite for quality management at the time of designing a machine learning based system is not available at the time of its operation, the overall logic of system quality assurance may collapse. This is the reason why the identification of available and reservable data is an important factor to examine an operational system.

6.9.2. Approaches

There are the two patterns described earlier depending on whether quality tests are conducted by a developer, etc. when a system is updated.

(a) Cases where the quality check process comes before updates

- In advance, estimate the frequency of updates or examine judging criteria for a necessity of updates.
- Analyze the update process required at the time of operation in the design stage and envision and analyze its outline to design specific procedures prior to the beginning of operation. It is necessary to take note of at least the following points.
  - How to collect available data from an operational environment and deploy such data in a developmental environment to be updated.
  - Preprocessing method of data for additional training and method of placing filters and labels.
  - Range of data used for additional training and model updating. How to

section is particularly regarded as “regression test”. However, “regression” is used in totally different contexts in the field of machine learning. Therefore, we intend to avoid the use of those terms in this document.

eliminate old data especially when environmental changes are expected due to the passage of time.

– Examine prior to the beginning of operation a method of quality tests at the time of updates, especially, judging criteria for acceptable updates (or a method of decision-making).
– It may be necessary to decide in advance how to handle some specific cases where the quality deteriorates depending on its application.

(b) Cases where updates are made without going through the quality check process in a developmental environment

– Envision in advance a possibility of notable quality deterioration caused by additional learning and the range of impact on systems if such deterioration occurs.
– When said impact can be unacceptable, examine a technical or operational treatment to accommodate risks for overall system caused by quality deterioration of learning models due to additional learning within the acceptable range and incorporate this treatment into operation. For example, there are the following ways.
  ➢ Technically limit the range of output values and use surrounding system components (software) to prevent any deviation from the envisioned normal range.
  ➢ Rewind learning and stop or suspend operation based on performance monitoring from the outside of an operational environment.
– If there is a possibility of monitoring the quality prior to updates in an operational environment and operational systems, such monitoring is deemed to be useful for quality management (see Figure 16).

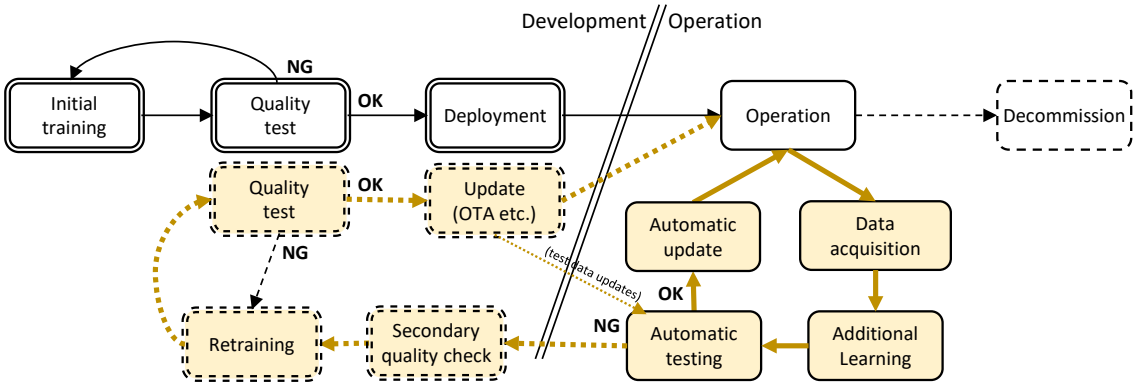

Figure 16: Possibility of automatic quality monitoring in operational environment

6.9.3. Requirements for quality levels

The relationship between required external characteristic levels and levels required for these internal characteristics are explained as follows:

- *Safety*
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 should be examined.
- *AI performance*
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- *Fairness*
  - AIFL 1: Lv 2 or above
  - AIFL 2: Lv 3 or above

The requirements for each required level for the above internal characteristics are as follows.

- Lv 1
  - Examine in advance how to respond to notable system quality deterioration caused by changes in external environment.
  - In the case where online learning is given, examine in advance the impact of unexpected quality deterioration and take measures from the system side such as the limitation of operation range if necessary.
  - When additional learning is given off-line, quality management in line with the previous seven paragraphs should be introduced.
- Lv 2
  - Monitor system quality deterioration and misjudgments by comparing with operation results within the range permitted by system use. It is necessary to sufficiently examine factors other than product quality such as privacy at the time of monitoring.
  - When online learning is given, regularly monitor additional learning results by any method. When any deviation from the requirements for performance is found as a result of monitoring, an immediate handling should be taken.
  - When additional learning is given off-line, conduct *regression tests on quality deterioration* with test datasets used in the system development stage to check if the quality has been maintained prior to updates. Update test datasets using the same method used in the system development stage where necessary.
- Lv 3

- Make sure to establish measures for monitoring system quality, including an operational system, compatible to privacy.
- When online learning is given, before the results of additional learning are reflected on systems, implement a mechanism to check quality to some extent within those systems so that updates are suspended if it becomes impossible to ignore unexpected quality deterioration. Make sure to ensure measures for making updates and modifications off-line.
- When additional learning is given off-line, the quality should be managed using data collected from operation, test datasets used for the initial system building and test datasets updated on a regular basis using the same method.

7. Technologies for quality management

7.1. A-1: Sufficiency of problem domain analysis

A problem domain analysis for machine learning (called *ML problem domain analysis* in this guideline) is a step in the development process. Diversity of data input to machine learning components is analyzed from various viewpoints in regard to the real-world situations, in which machine learning systems or services are used. In particular, the analysis aims to concretize the identified risks or divergence in system requirements as attributes of data employed for training and testing of the machine learning products. The analysis activities are basically similar to the risk analysis, hazard analysis, or problem domain analysis, all of which are subjects in Systems Engineering. These existing technologies, or the body of knowledge, can be helpful in conducting the ML problem domain analysis.

The ML problem domain analysis, viewed from Software Engineering, is a part of initial stages of software development processes, in which software engineers often rely on trial-and-error styles of the development. In this Guideline, the steps are basically meant to establish a piece of information necessary to achieve required quality levels of machine learning products and require activities such as what data scientists conduct in developing PoC systems with their informal know-hows concerning with the quality aspects of machine learning products.

7.1.1. Initial hints

Recent standard textbooks on Software Engineering (e.g., [96]) may present an overview of the analysis methods/steps that are employed in software development processes. Because safety analysis, or risk avoidance, is one of the primary foci in the system analysis steps, several modeling notations or methodologies have been proposed and studied. They include Causal Loop Diagram and STAMP/STAP [129]. Furthermore, the methodologies are equipped with the other detailed notations such as Fault Tree Analysis, Fault Mode and Effects Analysis, Loop Diagram, Feature Tree. In this guideline, such results are concretized as characteristics of data used to build, in particular, machine learning components. Therefore, this guideline recommends that Feature Tree is suitable as the modeling tool for the final artifacts obtained using the other modeling notations.

There might be two major problems if the analyses follow the mentioned approach; (a) modeling of input incidents, especially of risk factors, (b) design of the problem structure to be concretized as data characteristics. This section mainly refers to those concepts specific to machine learning based systems and machine learning components from these two viewpoints.

7.1.2. Modeling of risk factors in input space

Identifying risk factors mostly involves trial-and-error activities without any systematic means, and thus may often be conducted through brainstorming sessions by various stakeholders. It is, indeed, an analysis process regarding to Known –Unknowns. How well the risk identification is done is difficult to exhibit with quantitative measures, but may be confirmed only through the inspection by human experts.

As an example of such brainstorming activities, NASA Hazard Analysis Process, published by the National Aeronautics and Space Administration (NASA) [54], presents an outlined list of what should be considered in the early stage of the hazard analysis.

1) Standard Hazard List
2) Histrical experience/documentation from legacy systems
3) Your engineering training and experience

This literature also presents the "NASA Generic Hazards List" consisting of 25 items as the initial standard hazard list. Although these items include some hazard causes specific to particular applications, not causing any problem in the others, this list can be general purpose as the causes of hazard are concerned with mechanical operations of systems operating in outdoor environments. The list serves as a starting point for analyzing other systems.

Moreover, 2) is similar to what data scientists conducted in the past in order to build machine learning based systems. Since findings on similar systems in the past or early versions of the same systems are valuable, it is desirable to be utilized actively. In addition, a PoC system is recommended to act as a test-bed so that it helps the PoC developer understand those characteristics of final machine learning products. Finally, 3) can be considered as a form of utilizing what is called *domain knowledge*. It would be desirable to be incorporated in the body of knowledge for the users as well as the vendors involved in the machine learning software business. The NASA publication promotes to conduct the analysis activities based on these viewpoints, and then summarizes a conclusion as "*[Hazard analysis] [r]equires rational to justify hazard classification*".

This guideline recommends that developers or engineers conduct the modeling of risk factors basically from the above three viewpoints together with those findings collected from the PoC stage, and record precisely the modeling process and modeling results as well. Specifically, the followings are compiled into an integrated list of identified risk factors.

- Utilization of basic knowledge on the existing functional safety design, the existing hazard lists in each application domain, or brainstorming results based on the above NASA Hazard List;
- Utilization of analysis cases on prior systems and similar machine learning based systems;
- Introduction of domain knowledge by brainstorming with users (e.g. entrusters in

contracted developments); and
– Knowledge about data employed preliminarily and exceptional cases identified in the trials of the training in the PoC stage.

7.1.3. Design of problem structure as characteristics of data

This stage of the development process produces, as an output artifact, a piece of information on the attachment of attributes to data used for machine learning. It is affected inevitably by the characteristics of the next process, machine learning and training. Considering the training process that follows, the main activities of the process is to identify *attributes which are handled distinctively throughout the machine learning process,* which consists of the following two aspects.

4) Problematic if data are not recognized as distinct with respect to the differences in the output result values or risks; and

5) Problematic if data, resulting in the same output results, are likely to be understood as different due to their input values and characteristics of machine learning models.

4) is drawn mainly from the system specifications and the hazard analysis. A difference in output values is a difference in labels of training data, and risk incidents with different levels can basically be enumerated once the cause of hazard is identified.

On the other hand, 5) is dependent on application domains, for example, depending on differences in background images or character fonts for image recognition tasks. Moreover, 5) may be affected by preprocessing or architecture network of machine learning models. It is necessary to forecast a final implementation form to some extent in order to conduct such an analysis. This is called *Implementation Forecast* in software engineering because the information relating to the final implementation is indeed necessary to conduct the analysis at the early stage of the development. That may result in distorting risk analysis if carried out inappropriately, but it is inevitable for the quality management of machine learning components. Specifically, the activities are concerned with the implementation consideration made in the PoC development stage, and the analysis of PoC behavior using data.

In order to extract viewpoints with which the forecast is actually carried out, the three points listed in the previous section are considered applicable in view of *a risk that AI built makes different judgments in similar situations*. In the future, it would be desirable to combine 2) and 3) in the previous section and 5) in this section to accumulate system analysis results for each application area to some extent and compile them as detailed standard hazard lists in each area.

7.2. A-2: Coverage of distinctive problem cases

7.2.1. General

The internal quality in this section has two purposes, which follows the analyses described in Section 7.1. First is to prepare data needed at the time of training/testing so as to avoid *an unknown situation which a machine learning based system has not been learnt nor tested*. Second is to either decimate or integrate systematically an enormous number of combinations of prepared attributes so that building machine learning software of a required quality level is practical.

If a system intends to solve a simple problem and the number of combinations of attributes is a few dozen, it is possible to consider the amount of data that achieves a certain recognition level for each attribute. (Example: Only 10 characters x 2 fonts need to be recognized in number recognition and there is no need to take into account differences in paper quality. Then, there are 20 combinations.) If each attribute has such an amount of data, sufficient and comprehensive training can be practiced. However, since the number of combinations of attributes often exceed several tens of thousands if calculated in a naive manner, it is not practical to prepare data representing all combinations for testing. In a certain case, training data or testing data may be a few or even null as a result of breaking up attribute combinations. It would then be impossible to rely on the Law of large numbers to make a judgment such as the convergence of training.

Combination testing is developed in conventional software engineering for mitigating the complexity problem. The method, employing the notion of *test coverage*, establishes the degree to which combinations of attribute are detailed and the degree to which combinations of test condition are covered, yet devises them separately. The method is often used to evaluate test data in the conventional software testing, and is also expected to be effective for the case of machine learning, because training is data-centric. For example, [43] presents examples to apply the combination test technology (t-way) to evaluate dataset for machine learning.

7.3. B-1: Coverage of datasets

Establishing coverage of datasets is an extremely difficult but important problem in quality management of machine learning. A dataset, if a certain anomaly is associated with it, may derive trained machine learning models to exhibit unstable or unfavorable functional behavior in a certain specific envisioned situation, Lowering fairness levels.

As the problem domain analysis is finished, this section focuses on the issues with small differences in the dataset. The differences are so small that they are easily overlooked at the time of constructing the dataset. The discussions in this section are mostly related to the preliminary stages of data preparation or of data evaluation in the development lifecycle,

auxiliary checks may be sometime conducted in the software testing stage as well.

7.3.1. Plan for data acquisition

A data acquisition plan in the data preparation stage is a basis to study the issues relating to the coverage of dataset. In order to construct a dataset that are unbiased in view of operation time circumstances, the plan must refer to the range, period and size of data collection. Specific plans should be considered for each application following outcomes of the brainstorming sessions, which may be those activities, in the stage of analyzing *domain analysis problem*, to utilize prior knowledge about the similar applications so as to examine in advance the degree of diversity.

7.3.2. Pre-flight tests in data scrutinization stage

Auxiliary checking of data distribution belonging to certain attributes is desirable in some cases. These attributes are what, although identified as candidates in the early stages of the analysis, may not be adopted, or what may become latent, because they are composed to be a new attribute. The composed attributes are accompanied with explicit specifications, and in some cases those checks may be conducted mechanically, because data labels are given.

This is not perfect, because attributes overlooked in the brainstorming sessions cannot be recovered. It is worth considering.

Moreover, for some data, what is called data specific feature may be identified in the preprocessing stage, which makes it possible to check the distribution.

7.3.3. Additional tests in testing stage

In a case where the data distribution itself has some problems, what can be re-checked in the test stage is limited. For instance, there may be latent correlations between the adopted features and overlooked unidentified ones. In such cases, analyses of correlation or influence between the input values and their resultant counterparts output from the trained machine learning model may be helpful to see whether the model behaves as expected. The model may make inferences based on features different from what are assumed. It is worth considering this analysis when extreme safety is a major concern.

7.4. B-2: Uniformity of datasets

The uniformity of datasets is equivalent to considering the coverage of datasets, described in Section 7.3, as a single *case* of all the input data. The measures listed in each section of 7.3

are deemed applicable here as well.

7.5. B-3: Adequacy of data

7.5.1. Quality Control Cycle from the perspective of data

This guideline mainly examines the lifecycle process from the perspective of the development of machine learning models, but from the perspective of data quality, the lifecycle process of establishing consistency between data collection policies and actual data is also important. From the standpoint of data, there may be a recursive process iterating the following steps.

1) Determination of data collection policy
2) Collection and selection of data in accordance with the policy
3) Validation of both the policy and data

In (1) determining the data collection policy,

- Formulation of a data collection policy that is appropriate for the quality of the target machine learning system
- Advance consideration of the acceptable range of data validity
- Determination of the data validity evaluation method in advance

are necessary.

In the process of (2) data collection and selection, we perform

- Determination of the appropriateness of the selection and collection method,
- Process management of the actual selection and collection methods,
- Collection and consolidation of evidence on selection and collection methods,
- Management of meta-information about the source and traceability of data

in accordance with a defined policy.

In (3) validation, we perform

- Verification of data in accordance with the policy,
- Review of the explanatory nature of the policy itself,
- Feedback to the policy based on the collected data,

And the process continues back to (1) until a sufficiently satisfactory and valid policy and data pair is obtained. Furthermore, this entire process is subject to quality management and assurance, such as:

– Evaluation of the completion of the process itself,
– Evaluation of the skills of the people in charge of executing the process and their appropriate assignment,
– Evaluation of the mechanism for managing the entire process and securing of evidence,
– Evaluation of the system for managing the entire process and ensuring its traceability.

In relation to the development lifecycle described in Section 1.8, such a cycle of data policy improvement is abstractly organized as a repetition of trial and error in the PoC process, but in reality, sometimes the improvement cycle described above is repeated even in actual development process, leading to agile developments. In such a case, it is particularly important to assess whether the requirements definition has been updated together with the data collection policy, or whether it is necessary to reconfirm the contents identified in the previous stages (internal quality A-1 to B-2, etc.) in accordance with such updates, in order to ensure the overall quality. It is also extremely important to ensure that such policy and requirement updates are subject to change management in order to reconfirm and verify quality later on.

7.5.2. Technical support for organizing outliers and corner cases

There are several techniques that can be applied to efficiently extract and organize data that require data validity judgments either as outliers or as rare cases to be addressed.

First, DeepXplore [101] uses the Neuron Coverage technique for corner case extraction. Surprise Adequacy [73] provides a measure of the *surprise* of the inputs to a machine learning model, which also allows corner cases to be extracted from the input values. Tinghui et. al [99] improved on Distance-based Surprise Adequacy to perform corner case analysis. Whether the data extracted in this way should be removed as outliers or treated as important rare cases is to be determined on a case-by-case basis in conjunction with policies. Sec. 5.3 in Zhang [126] and Sec. 2.3 in Dong [56] are also helpful.

7.6. C-1/C-2: Correctness and stability of trained models

Checking the correctness and stability of machine learning models is a task corresponding to *unit testing* in conventional software testing. It is because the check is focused on each machine learning component. In this section, we will introduce the following topics.

- Basics of software testing techniques [41], the aspects to be taken into account depending on the characteristics of machine learning components, and recent research works on machine learning component testing

- Potential of technology for testing of stability directly

7.6.1. Testing machine learning components

7.6.1.1. Characteristics of test target

First, there are two distinctive test targets to consider when discussing the quality of machine learning component results: training and learning programs that input training datasets to obtain trained learning models, and prediction and inference programs of which the resulting trained learning models specify functional behavior.

Second, there is no clear criterion for deciding whether a computation result is correct, which is called test oracle problems [44]. For a training and learning program, it is difficult to know in advance the correct value of its computation result, that is, a trained learning model. In addition, the answers derived by prediction and inference programs are not definite but accompanied by probabilities referring to certainty degrees, which illustrates difficulties to define absolute criteria for correct answers [57]. In general, machine learning programs are classified as non-testable programs [121].

7.6.1.2. Test Oracle Problems

Software testing technology consist of test input generation methods and test execution result confirmation methods. The first is how to generate test inputs that efficiently achieve the testing purpose. As we will see later, there are some interesting research works on test case generation methods for machine learning components.

The second is concerned with methods for comparing the results of program execution with known correct answers to a given test inputs and is collectively referred to as test oracle techniques. A non-testable program is one in which the correct answer to a given test input is not known, and derived oracles, pseudo-oracles, or partial oracles have been used in software testing so far.

Metamorphic Testing (MT) [50] is a practical method of partial oracles. It was introduced as a testing method for programs in which it is difficult to know in advance the correct value of the computation result for an input, such as numerical computation or translators involving compilers, and was later applied to the testing of operating systems and security systems to use implicit oracles [51]. It is now a standard testing methodology for machine learning components [92]. For some testing purposes, a combination of partial oracles and statistical testing can be useful [94].

7.6.1.3. Two Perspectives on Testing

In general, when checking a program, positive testing and negative testing are conducted.

Positive testing is to confirm that the test target exhibits the expected functional behavior.

For the sake of explanation, let's assume that a program has pre-conditions and post-conditions. The relation is established such as *when the pre-condition is satisfied, the post-condition is satisfied in the state after the program body is executed*. When the above relationship holds for test input data that satisfies the pre-conditions, the program under test is considered to pass the test.

Negative testing checks whether a program will not fail catastrophically under unexpected conditions. In general, a program is expected to behave reasonably even in response to input that does not satisfy its pre-conditions. For example, the program should not run out of control and terminate abnormally. Checking these cases does not need an application-specific correctness criterion. Such test oracles are called implicit oracles, meaning that there is no explicit test oracle at all.

Fuzz Testing or fuzzing [88], which uses randomly generated data, while satisfying some constraints, as test input, has been known effective for integration testing of open system software such as operating systems or security systems. This method is sometimes used in negative testing of programs. An appropriate fuzz, a test input data, is generated to meet each test objective.

In testing for machine learning components, positive testing is, in a sense, related to checking the accuracy. We can use data that follow the same statistical characteristics as the training dataset. On the other hand, negative testing includes checking the accuracy in regard to the generalization performance and checking behavior against unexpected inputs in the case of checking the stability or the model robustness. Stability testing includes the case where the input is either corrupted data or adversarial examples. As described below, various fuzzing techniques have been proposed for testing machine learning components, depending on the purpose and perspective of the test.

7.6.1.4. Test Coverage Metrics

In machine learning component testing, it is important to know an extent to which a target trained learning model becomes activated for a given test input. The idea corresponds to the test coverage criteria in conventional software testing methods [41]. Neuron Coverage (NC) [101] is, an early proposal for such a structural metric, defined by a frequency of activated neurons. Later, in order to use more detailed information than NC, several coverage metrics are proposed [84]; those are based on structural patterns over activated neurons, such as correlations between activated neurons in different layers. If these structural coverage values are large, it can be assumed that a wide range of the learning model is activated, and thus is checked as well. The test coverage is considered sufficient and thus we may conclude that such test inputs are useful.

In general, it is difficult to define the test inputs used for negative testing which do not satisfy pre-conditions, because the statistical characteristics of training data, which might be

regarded as such pre-conditions, are not explicitly stated for machine learning components. Surprise Adequacy [73] provides ways to check how test inputs deviate from the statistical characteristics of the training data. Although the adequacy metrics are originally introduced for evaluating quality of test inputs, they are also considered to be a kind of indicators referring to degrees of being outliers. Note that these indicators refer to the internal activated states of trained learning models, and thus that measuring these requires monitoring of execution states of trained models.

7.6.1.5. Automatic Generation of Test Inputs

In general, it is important to establish a method on generating appropriate test input data for each test target. In addition, such an appropriateness depends on the testing purpose, either positive or negative.

When the test target is a training and learning program, the input is a collection of data (a dataset), and the notion of dataset diversity [90] provides a guideline for test input generation methodologies for both positive and negative testing.

In testing of prediction and inference programs, various types of data that were not considered during training stages are expected to input for checking prediction results. The method is sometimes called Test-time Augmentation. For generating test data, there have been various approaches including those to adopt machine learning techniques. They include conventional Data Augmentation methods for image data [77], or methods using Generative Adversarial Networks (GANs) [63]. Furthermore, some methods make use of the adversarial example generation method [95] [128]. In the case of generating a large number of test data, the aforementioned dataset diversity is also useful as a conceptual guideline.

7.6.1.6. Test Generation and Coverage Metrics

In generating test inputs, it is desirable to find useful data that can be used to conduct testing efficiently depending on test purposes. As in conventional software testing techniques, we apply some metrics to control the process of generating data that, for example, improves the structural coverage criteria. This method is collectively called Coverage-Guided Test Generations. There are some research works that use the NC when machine learning components are test targets. The NC is combined with the classical data augmentation method [113], or with the GAN-based data generation method [127].

Some recent works report that the structural coverage may not be useful depending on the purpose and perspective of the test [65]. For example, the correlation between NC and adversary robustness is known to be small [118]. The NC values are more highly correlated with model capacity than with defects in trained learning models [133]. For example, experimental results [92] show that the accuracy can be high even if the NC is small, and

conversely that the NC value can be large even when the accuracy is poor. In addition, it is known in conventional software testing techniques [67] that the structural test coverage indices have a small correlation with the effectiveness of test suits or the efficiency of detecting faults. This empirical report results seem to be applicable to machine learning components as well. As alternatives to the structural coverage criteria, a new method is proposed [118] that is based on an error function to be used as metrics.

Although identifying data for executing corner cases is desirable for effective testing, the relationship between inputs and corner cases is not clear in machine learning components. Corner case test data, for conventional programs, are to check program locations of decision points, each of which leads to different final outcomes. Because such decision boundaries are difficult to be identified in trained machine learning models, it is not clear what test data go through such corner cases. Note that corner cases are not necessarily the same as outliers, because the two mentions data characteristics from different perspectives; the corner cases are in relation to the decision boundary in trained machine learning model and the outliers are usually defined in regard to data distribution of the dataset.

Last, viewed from a slightly different point, NCs can alternatively be regarded as a simple indicator of the activated neurons [91]. In fact, experiments on testing of training and learning programs are reported to use statistical indices calculated from NCs or collections of NCs as test indices [133][94].

7.6.1.7. Prioritization for Preparing Training Data

In conventional software testing, regression testing involves a large number of test cases. Selecting test cases that are useful for detecting faults as early as possible can improve the overall efficiency of test activities. Therefore, when a large number of test cases are given, prioritization is performed [104], where choosing metrics to find useful test cases is essential.

For machine learning components, prioritization ordering techniques can be used from the perspective of improving efficiency of the data preparation work, especially of work for labeling [46]. We consider here the task of adding new test data to an existing training dataset when a lot of data without answer labels is available. First, candidate data are picked up in a certain way to be put into a data pool. Labeling data selected from the pool with correct answer tags is done manually, which is time-consuming to require a lot of human work if the number of data is large. Therefore, if we select data that exhibit characteristics different from the existing training data and label only these data, we can obtain useful new data while reducing the total amount of work.

Technically, it is a method of deciding whether data is useful, and it comes down to another problem of devising appropriate indices to be used for the decision. Recent research works propose a number of methods using either external metrics [60] such as Confidence and Gini Impurity, or internal metrics [46] such as Surprise Adequacy [73]. Choosing metrics should

take into account what kind of failures are to be detected or what different features are desirable in relation to the existing training data.

It is interesting to see that the prioritization method, used for selecting test data in conventional software testing, can be applied to the task of training data preparations in the case of machine learning components.

7.6.2. Technologies on stability issues

The technologies on stability issues are broadly divided into the evaluation and improvement. Figure 17 illustrates how each technology is related to the level described in Section 6.7.3.

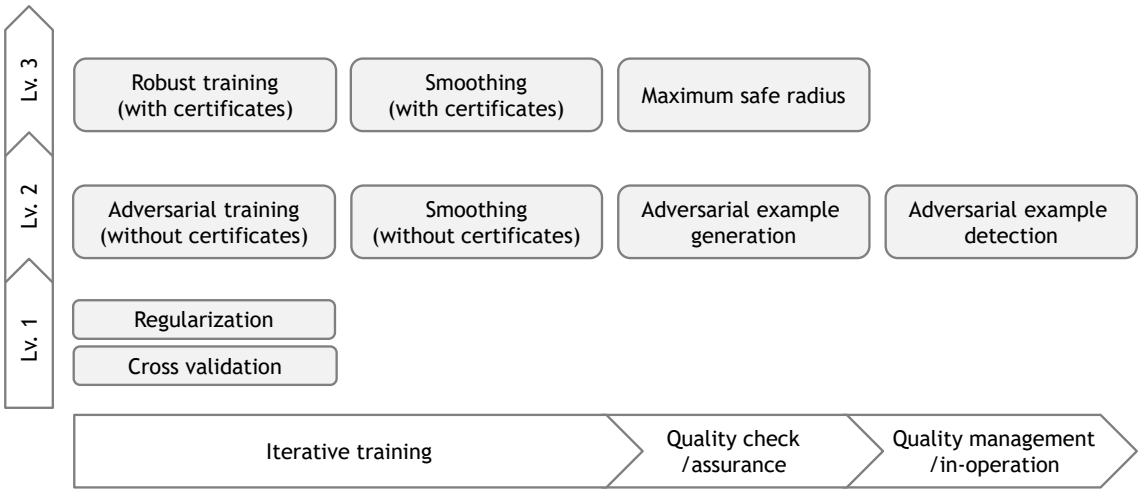

Figure 17: Technologies to evaluate and improve stability (process and levels to which technologies are applied)

7.6.2.1. Cross validation

Cross validation is a classical method to mitigate the over-fitting problem, and thus to improve stability levels. The idea is simply dividing a whole dataset into training, validation, and testing datasets [74]. For example, in K-division cross validation, a whole dataset is divided by K, and $1/K$ of the dataset is used for validation and the remaining $((K-1)/K)$ datasets for training. The roles of datasets, either training or validation are interchanged. Cross validation is recommended to Lv 1.

7.6.2.2. Regularization

Regularization is a methodology to mitigate the over-fitting problem, and thus to improve stability levels. Especially, regularization methods suppress the absolute values of learnt weight parameters not to become excessively large [97]. For example, loss functions may include a regularization term to increase as absolute values of parameters become large). Dropout is another regularization method [111], in which neurons are randomly excluded from the training target during the training process. Dropout may obtain the same effects as cases where several machine training models are trained simultaneously. Regularization is recommended to Lv 1 if the stability is an issue.

7.6.2.3. Adversarial example generation

Adversarial examples are those data that cause miss-inference in machine learning components. Such data are augmented with a slight perturbation, a semantic noise, that human eyes are not able to recognize. Various methods to generate adversarial examples have been proposed [101][48]. A stable machine learning component is expected not to cause miss-inference even when the perturbation with the input adversarial data is large. Therefore, the degree of perturbation is used to evaluate the stability. Unfortunately, no practical tool to generate such adversarial examples have been established, this technology will be hopefully applicable to Lv. 2 evaluations in the future.

7.6.2.4. Maximum safe radius

The maximum safe radius refers to the minimum distance between the original data and its adversarial example. Adversarial example generation (7.5.2.3) looks for nearby adversarial examples, while the maximum safe radius guarantees that there is no adversarial example in the neighborhood (inside the hyper-sphere of the maximum safe radius). Methods to accurately calculate the maximum safe radius are proposed in [71][114] that use SMT solvers. However, a cost of accurately calculating the maximum safe radius is so high that the size (number of neurons) of networks is limited. Alternatively, methods to approximately calculate a safe radius smaller than the maximum safe radius are proposed in [120][45][123]. Moreover, a method to approximately calculate a radius that probabilistically guarantees safety (though not 100%) is proposed in [119]. These technologies are still in the research stage, but they can guarantee that there is no adversarial example inside the hyper-sphere of the maximum safe radius. Therefore, they are expected to be applicable to Lv.3 in the future.

7.6.2.5. Generalization error bound

The generalization error is the expected value of the incorrect answer rate for all input data (not only for a particular data set). In general, it is difficult to measure the rate of incorrect answers because of the large number of input data. However, it is possible to estimate a probabilistic upper bound on the generalization error which *guarantees with probability p % that the generalization error is less than e %* [116]. At present, the method is still in the research phase and the accuracy of its estimate is not yet high. However, it is expected to be applied at Lv.3 in the future as a measure of generalization performance.

7.6.2.6. Adversarial training/robust training

Adversarial training is a learning method, which looks for neighborhood data that is likely to cause miss-inference [86]. Compared to the standard training method, it mat be able to improve the resistance of trained machine learning models against adversarial examples. Although this technology is still in the research stage, it is expected to be applied to Lv.2 in the future.

On the other hand, a robust training method is to eliminate neighborhood adversarial examples [122]. An approximate calculation method of the maximum safe radius is given together to guarantee that adversarial examples do not exist near each data in the training dataset. Although this technology is still in the research stage, it is expected to be applied to Lv.3 in the future.

7.6.2.7. Randomized smoothing

Randomized smoothing is a method to augment data with randomized noise so as to calculate the final result value to be a mean with respect to the noise distribution. The method is reported to improve the stability against the L2-norm adversarial attacks [83]. Furthermore, methods of adding randomized noise are proposed so as to guarantee the correctness of inference results [79][53]. In these methods, the randomized smoothing is applied to the neighborhood of input data. Because the statistically expected value is approximated by an average, multiple running results are needed. Furthermore, the stability degree is increased as the randomized noise is large, which in turn lowers the correctness. Such a trade-off relation must be taken into account. Nevertheless, this technology is expected to be applied to Lv.3 in order to guarantee the stability of inference results.

7.6.2.8. Adversarial example detection

Adversarial examples detection is a method to check, at runtime, whether incoming data are adversarial or not, and is one of the active research areas [124][85]. Although such a technology is still in the research stage, they will be expected to be applied to Lv.2 for protecting potential adversarial examples in machine learning components.

7.7. D-1: Dependability of underlying software system

7.7.1. General

The dependability of underlying software system is an important item even in conventional software. Quality management is expected to be difficult especially in implementation of machine learning AI which uses a large number of libraries including open-source libraries. Open-source software does not usually count on guarantee. Therefore, in relation to end users, open-source users (developers and operators) are responsible not only for differences in malfunction but also for discovering and monitoring potential errors and, in some cases, making modifications.

Moreover, when a machine learning model is built, it has been revealed that a training process incorporates bugs (program error) so that the impact of bugs does not appear in tests, causing quality deterioration [93].

On the other hand, in relation to conventional software, configuration management and quality monitoring of libraries and fundamental software are emphasized and infrastructure and services for configuration management and quality monitoring are improving. Some data included in those services contain information such as libraries specific to machine learning, although it is somewhat limited. These existing infrastructures are deemed worth being utilized in applications of machine learning.

7.7.2. Quality management of open-source software

It is desirable for each business operator to think about how much open-source libraries can be trusted and their quality maintained by itself or outsourced.

It can be expected that supported libraries whose quality is assured and software that went through the quality inspection process are used where necessary in relation to a necessary quality level.

7.7.3. Configuration management and tracking of bug information

For example, common Platform Enumeration (CPE) [35] is used for configuration management of software components as a common ID to list system constituent components, in the security field. There are commercial products that manage versions of software components in use and extract information on their updates based on CPE. A list of vulnerability information related thereto called Common Vulnerability Enumeration (CVE) [36] does not include bugs that are not directly related to security vulnerability. However, at least tools related to CPE are very likely to be beneficial to track the latest version of libraries and infrastructure software.

7.7.4. Possibility of specific check thorough testing

Moreover, when constituting software components have any bug, that bug does not always have a direct effect on actual ML results in machine learning components different from conventional software. If software with bugs is placed in a feedback loop of learning or training, a trained learning model memorizes behaviors of said bug and training seems to be superficially successful in some cases. In these cases, it is reported that potential quality deterioration caused by software bugs can be found by analyzing statistical behaviors using any of the metamorphic test technologies listed in Section 7.6.1.

7.7.5. Software update and possible adverse effects on performance and operation

On the other hand, actual software whose configuration is complex often operates unintentionally even if a developer of software libraries updates it with the intention of improving performance and operation. Depending on how a machine learning based system is configured, behaviors of bugs are sometimes adapted or learnt in the training process. Therefore, when any software constituent component is updated, its operation and performance must be reviewed. In some cases, it is important, to make a judgment to start training over or keep using an older version taking into account vulnerability.

This Guideline cannot recommend any option, because a specific judgment depends on each situation. However, it is important to record the background of each judgment for accountability reasons in order to claim *the proper quality management*.

7.8. E-1: Maintainability of qualities in operation

This section describes technologies to maintain internal qualities satisfied at the commencement of operation throughout the operation period. In order to maintain internal

qualities realized at the beginning of operation in spite of changes in external environments that may arise during operation, a machine learning component needs to respond to those changes. As described in Section 6.8.1, there are two operational patterns therefor. One is to make batch-processing updates by returning to the developmental environment and deploying it again. Another is to automatically update the software component when needed or at a high frequency in the operational environment. In the former pattern, monitoring to determine the timing of update and update processing are main elements, while in the latter pattern, automated update processing is a main element. In order to realize this, on-line learning [107] is adopted. Even if updates are made automatically, it is necessary to monitor if automatic updates operate properly and process updates to respond to cases where there is any deviation from normal operation conditions.

Some of monitoring technologies necessary for both patterns are presented here. We focus especially on technologies to detect changes in data distribution over time called concept drift [62]. Moreover, technologies to retrain machine learning models which play the core role of updating trained machine learning models and technologies to create additional training data used therein will be presented.

7.8.1. Monitoring

The relation between newly acquired input data and output (inference) results may have changed at the time of operation from the relation between them at the time of training due to various factors including changes in external environments. When a machine learning model trained in the design or development stage using such data whose input and output relations change is kept using during operation, its performance (accuracy) may deteriorate and result in serious damage. Therefore, it is required to continuously monitor behaviors of machine learning based systems and machine learning components for the purpose of checking if the quality fulfilled at the beginning of operation is maintained throughout the operation period.

There are the following four tasks of monitoring during operation.

- Accuracy monitoring
- KPI monitoring
- Model output monitoring
- Input data monitoring

Accuracy monitoring directly measures the accuracy of trained machine learning models. This monitoring is divided into some patterns in accordance with the method of collecting correct answers compared with inference results of trained machine learning models required for calculating the accuracy. That is, after making an inference, (1) cases where correct answers are acquired automatically after a certain period, (2) cases where correct answers

cannot be acquired automatically so that it is necessary to manually put labels and (3) cases where manual labeling is beyond budget or it is impossible to put labels. It is necessary to select an appropriate monitoring method in accordance with the above grouping of cases where correct answers are collected in order to appropriately monitor the accuracy. Moreover, some applications emphasize not only monitoring the accuracies of models but also monitoring from the viewpoint of KPI, *KPI monitoring*, in line with a conversion rate and the benefit of users. In this case, it is required to monitor the consistency between the accuracies of the models and the KPIs of the applications.

Model output monitoring and *input data monitoring* refer to the monitoring of results of inferences made by a trained machine learning model and the monitoring of its input data, respectively. The monitoring methods are divided into human monitoring (100% sampling), automated monitoring (alert conditions are known) and filtering (conditions with higher possibilities of alert are known). Model output monitoring is further categorized into a case where each output inference is checked by experts as in the case of medical diagnostic and a case where all inferences are checked altogether after a certain period of time. In the same way, various conditions may be imposed in the case of monitoring by filtering (For example, false positive is acceptable but false negative is unacceptable). In addition, when input data is monitored, whether an alert is issued in the case of monitoring by filtering or input is disregarded depends on an application.

7.8.2. Concept drift detection methods

Concept drift is one of major causes of the deterioration of the accuracy of trained machine learning models during operation. A variety of monitoring (detection) methods have been proposed recently. Concept drift detection methods are categorized as shown in Table 4 in accordance with whether correct-answer labels on data acquired during operation are used [105].

Table 4: Classification and characteristics of concept drift detection methods

Use of label	Method	Characteristics
Labeled detection (Supervised detection)	Sequential analysis	Monitoring of absolute values such as accuracy
	Statistical Process Control	Monitoring of the increase or decrease of error rate (Early detection by finding indicator)
	Window based distribution monitoring	Monitoring of distribution differences between training time and operation time
Unlabeled detection	Novelty detection / clustering method	Simple detection by clustering

(Unsupervised detection)	Multivariate distribution monitoring	Detection by applying statistical hypothesis testing to data distribution
	Model dependent monitoring	Output dependent of models, such as confidence score

Studies of unlabeled detection methods have been carried out actively in recent years. For example, a literature [59] proposes an unknown class detection algorithm (MINAS) in multi-class problem based on clustering methods such as k-means. Moreover, another literature [103] proposes an algorithm of incremental KS test which improved the amount of calculation by developing incremental Kolmogorov-Smirnov test to judge whether samples are generated from the same distribution. Moreover, a literature [82] proposes a method of detecting data changes by using confidence scores associated with output from trained machine learning models without using labels (CDBD).

7.8.3. Retraining

When any change in data distribution or deterioration in the accuracy of a trained model is detected as a result of the monitoring describe above, it is necessary to retrain the machine learning models using datasets which add recent data or replaced recent data with existing training data. Many researches have been made about this retraining. For example, a literature [112] provides a platform design method to automatically determine the appropriate timing of model update in relation to existing machine learning frameworks. Moreover, it has also been proposed a method of applying balanced parameters of both *existing models* and *models that learnt new tasks* to retrain models in order to reduce forgetting of old tasks caused by retraining of a neural network called catastrophic forgetting [80].

7.8.4. Creation of additional training data

There are many cases where labels cannot be collected automatically. In this case, it takes much cost to manually place labels to new data acquired at the time of operation. A study on reduced costs of retraining by reducing the number of data to which labels are attached has been proposed through a method of reducing labeling work using software equipped with GUI (Graphical User Interface) [117] and active learning have been proposed to solve this problem [106].

8. Fairness

In this chapter, we analyze the social background and problems of quality management regarding fairness, as well as the technical issues specific to fairness. It also summarizes the issues that should be considered as internal quality characteristics to address fairness.

8.1. Background

8.1.1. Social demands and social principles

8.1.1.1. AI Social Principles, etc.

In recent years, there has been a lot of discussion about ethical and fairness issues of AI, and the norms in society of AI. Correspondingly, all the documents on AI social principles referred in section 1.9 explicitly address requirements for fairness and ethics as important topics. The Council for Comprehensive Innovation Strategy's "Principles for a Human-Centered AI Society" [20] states in the point 6:”Principles of fairness, accountability and transparency" that "under the design concept of AI, all people shall be treated fairly without unfair discrimination on the basis of their race, gender, nationality, age, political beliefs, religion, or other diverse backgrounds”, and requests the sufficient consideration regarding fairness from the design phase of AI system. The OECD AI Principles [21] request AI systems to be designed with human rights and diversity in mind, and to include appropriate *safeguard* mechanisms against the fairness related issues. Furthermore, based on the OECD AI Principles, AI fairness subjects have been mentioned in EU AI White Paper [24] and the European AI High-Level Expert Group's AI Transparency Guidelines [26] ,and is considered to be gaining consensus at the principle level [58][29].

8.1.2. AI Governance

According to a report by the Ministry of Economy, Trade and Industry (METI) [29], AI governance is: “the design and operation of technical, organizational, and social systems by stakeholders with the aim of maximizing the positive impact of AI while managing the risks arising from its use at a level acceptable to stakeholders”. The principles exemplified in the previous section are generally conceptual, technology-neutral, and cross-cutting statements of goals, while AI governance aims at achieving these goals through activities of the developers, supervisors, and regulatory enforcers, based on various grounds such as legally binding rules, non-legally binding guidelines, and international standards.

8.1.2.1. Legally binding rules

Fairness has a long history in the U.S. legal system and has been the basis for several concepts. More than half a century ago, the Civil Rights Act of 1964 was enacted to prohibit discrimination by race, religion, etc. Title VII of this act addresses discrimination in employment. Through the administration of these laws, two important legal concepts, Disparate impact and Disparate treatment, have been established. However, these laws and other historic laws do not address issues specific to current AI system technology. For example, Facebook's 2019 lawsuit was [9]also based on technology-neutral laws that are not limited to AI.

Recently, in April 2021, the EU announced an AI-specific policy package, including a regulatory framework draft on AI [22]. There has been a lively debate among experts that legal regulations that apply to a wide range of AI fields may be a significant impediment to innovation. In terms of the severity of the restrictions, the EU's draft regulation takes a pragmatic, risk-based approach to classifying AI systems according to their intended use. For example, all AI systems those may contravene EU's values, for instance by violating fundamental rights will be classified as *unacceptable risk AI* and prohibited in principle. AI systems that create a high risk to the health and safety or fundamental rights of natural persons are classified as *high-risk AI* and subject to compliance with certain mandatory requirements and an ex-ante conformity assessment. As with the case for GDPR, when the regulation is formally passed after further deliberations, it is expected to apply not only to AI system providers within the EU, but also to AI system providers outside the EU if used within the EU.

8.1.2.2. Non-legally binding guidelines

In general, it can be assumed that non-legally binding guidelines serve two purposes.

I. In cases where legal regulations exist, to specify the means to actually fulfill the requirements (as detailed checklists, etc),

II. In cases where legal restrictions do not apply, such as low-risk classification case, provide the basis for ensuring the quality of the AI products and services.

For example, "The Assessment List For Trustworthy Artificial Intelligence (ALTAI)" [27] by the European High Level AI Expert Group addresses fairness as a "Diversity, Non-

[9] In March 2019, the U.S. Department of Housing and Urban Development filed a lawsuit against Facebook for violating the Fair Housing Act when its targeting ads excluded certain demographics from ad delivery based on race and other characteristics.

discrimination, and Fairness" requirement, and lists the following checklist items as the purpose of *avoiding unfair bias*:

- Did you establish a strategy or a set of procedures to avoid creating or reinforcing unfair bias in the AI system, both regarding the use of input data as well as for the algorithm design?
- Did you consider diversity and representativeness of end-users and/or subjects in the data?
- Is your definition of fairness[10] commonly used and implemented in any phase of the process of setting up the AI system? (Did you consider other definitions of fairness before choosing this one? etc.)

This guideline can also be utilized for both the purposes I and II mentioned above.

8.1.2.3. International standard

Reports and standards on ethics, transparency, and fairness of AI are being developed in ISO, IEC, and IEEE. These international standards are described in section 10.2.

8.1.2.4. Inclusiveness

Addressing fairness is also essentially *achieving diversity*, and inclusive development, design and deployment that involves various communities from the beginning of the development process could help prevent further social damage and reduce existing social inequalities. An international multi-stakeholder initiative (Global Partnership on Artificial Intelligence) was established in June 2020 to achieve responsible, human-centered AI development and use, including this *inclusiveness*.

8.2. Ethics and Fairness definitions in this Guideline

At present, we can hardly find out clear-cut definitions of terms such as *ethics* and *fairness*, which are fully agreed in both the social and technical domains. In this document, we define *ethics* as *good behavior in the context of real-world norms* as a property of the field of sociology. This ethics is not limited to fairness but may also include policies on deterring certain forms

[10] *Fairness* here is roughly equivalent to the *fairness metrics* of this document.

of judgment, certain kinds of safety consideration, and on dilemmatic decisions. In general, ethical requirements for systems are implicitly presented by society, are derived from social norms that are not always explicitly stated, and serve as constraints and considerations for defining requirements mainly in the planning and design stages of systems.

On the other hand, we narrowly define *fairness* as *not being treated differently, on some defined basis, due to differences in the inputs that are not defined as requirements*. In discussions at the level of social norms, this definition of fairness is considered to be part of ethics.

The fairness of the machine learning elements that this document aims to achieve is defined further narrowly with engineering focus, as a set of identification requirements that are subject to specific fairness guarantees, and that are discovered at the requirements definition stage from requests for ethics, functionality, etc., and are derived by the system provider as specific requirements from the requirements definition.

8.3. Difficulties with fairness

8.3.1. Diversity of requirements

In the previous section, we described the fundamental definition of fairness in this document. However, there are multiple perspectives to explain, assure, and convince people of *being treated equally*, and often, just saying *being fair* is not enough. A fair system would be developed only if fairness is defined in a detailed and mathematical way. In order to make such investigations on fairness, it is crucial to understand the key perspectives of fairness, especially related with various difficulty aspects.

In this section, we will focus on the following two discriminations that have been often referred in the subjects related with the equal employment legislation of the United States, and other discussions about fairness.

(A) Disparate treatment discrimination

(B) Disparate impact discrimination

Disparate treatment discrimination refers to cases where there is some unfair treatment in the process, where fairness is intentionally compromised, and is considered a basic requirement in the employment process.

On the other hand, *Disparate impact discrimination* refers to cases where resulting fairness is compromised, and is sometimes regarded as a problem, such as in the adverse events to minorities. The definition of resulting fairness needs to be defined in detail using target indicators, such as *equal hiring result number*, *equal ratio of hiring to applicants* (between men and women). There are cases where deciding the right fairness index for the ethical requirement require quite a consideration. Furthermore, it is often not enough to simply *not include intentional discrimination in the process*; it may be necessary to intentionally introduce *discriminatory treatment* in order to eliminate discrimination as a result, and this may even

involve an intentionally unequal process called affirmative action. From the perspective of engineering for society as a whole, this can be thought of as a situation where feedback control with overshooting is applied toward a target value that is considered to be correct, while from the perspective of AI system implementation, a specific output distribution is required as a functional requirement based on ethical and fairness considerations.

Regarding the terms *equal opportunity* and *equal treatment* that are often referred to in Japan, there exist multiple levels and need to be clarified case by case. Taking *equal treatment* as an example, it can refer to *strong equality*, meaning that measures are taken to ensure equality (e.g., algorithms) as a system during the development stage, or it can refer to *weak equality*, meaning that intentional inequities are avoided during the development stage. Especially in the field of statistical machine learning, the latter type of equal treatment alone is often insufficient to ensure fairness due to various factors. In addition, some situations such as personnel evaluations, may even require a *justification* that is felt by the individuals concerned that are hardly quantitively defined.

(Reference) Group Fairness and Individual Fairness

In general, fairness demands are concerned with attributes that may cause inequity such as race and gender (attributes requiring consideration). Group fairness is to avoid discrimination among different groups with respect to some attribute value (e.g., disadvantageous treatment of *women* which is a value of the attribute *gender*), while individual fairness seeks for similar individuals to be treated similarly. As described in the following sections, most of the current general-purpose machine learning metrics and measures are based on the premise of group fairness that mainly addresses *attributes that require consideration, generally called sensitive attributes*, and this guideline also assumes the group fairness perspective unless otherwise noted.

In the case where *justification from the individual person point of view(=individual fairness)* is required, such as personnel evaluations, it is difficult to define general metrics thus measures to satisfy the requirement must be considered for each system correspondingly. As for individual fairness, research on *degree of similarity* using distance learning [66] for example, have been proposed, and we hope to see more of this in the future.

8.3.2. Ambiguous social demands for Fairness

When the demand for fairness in the system is caused by the requirements of laws and regulations (even if partially), the concretization level gap between the legal requirements and the technical implementation also becomes an issue. For example, when laws and regulations stipulate that *no disadvantageous treatment shall be given*, it is not always obvious what acts or omissions in automatic processing by machines and their design is considered as disadvantageous treatment. Furthermore, whether *equal treatment* demands on the process

how people build AI or on the AI outcome can make a big difference in actual development work. As norms and best practices such as this guideline become more widespread, it is expected that something like a market view of fairness measures will be formed eventually, but at the moment, it is strongly required to be able to properly explain *how you thought and how you implemented it*.

In addition, the requirements of laws and regulations are not always self-evident as to what is the object to ensure fairness (in other words, *what would be values for the sensitive attribute?*). Sometimes it is explicit, such as equal opportunity for *men* and *women*, but more often it is an incomplete enumeration, such as "gender, race, etc.". Even in explicitly defined case, consideration and explanation of the implementation policy will be needed, if the attribute is not necessarily included in the input data, or if the attribute value is subject to estimation and can contain measurement errors.

8.3.3. Embedded inequities in society

When building machine learning elements from data, it is important to consider the case where the data itself, such as training datasets, may contain inappropriate biases. Data obtained from the real world may sometimes inherit the existing biases contained in society.

Inequity embedded in society is one reason why a fair system cannot necessarily be created simply by eliminating sensitive attributes through the development process. Therefore, in developing a system that uses machine learning, where fairness is important, fairness requirements should be clarified not only from the deduction of the data itself, but also from an (top-down) analytical perspective, leading to scrutiny at the data preparation stage. Such activities will also lead to clarification of social demands for the system that are not always specifically stated, also improving its accountability.

> (Reference) COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) Case Study
>
> It was pointed out that the COMPAS system, which is used to estimate recidivism rates as a basis for making decisions on parole for imprisoned prisoners in the United States, may be making significantly unfavorable decisions for black prisoners, and that socially embedded biases such as bias in crime detection may be included in the training data. Whether the point was valid or not depended on how the fairness metrics were defined, which also suggested the importance of defining fairness metrics.

8.3.4. Hidden Correlations and Proxy Variables

When building machine learning elements from input data that has a huge number of attributes and information, other attributes that seem unrelated to discrimination, or features of the input that are not clearly identified as attributes, may have hidden correlations with

sensitive attributes, resulting in statistical reproduction of discrimination. For example, such correlations between attributes such as name and gender, school name and gender, address and income, may result in unfair output even if the direct contribution of the sensitive attributes to the output is removed. It is also possible that unfair output may be produced as a result even if the direct contribution of the attribute to the output is removed.

In such a case, even if you try to remove social inequities by data synthesis as described in the previous section, you may not be able to synthesize training data that is appropriate in the real world. For example, if the name of a school has characteristics such as boys' school and girls' school, adding data by artificially synthesizing data with only the gender of the input data reversed will not result in valid data.

8.3.5. Variables requiring careful consideration

When trying to eliminate unfairness in the AI development process, it is sometimes difficult to decide how to treat attribute values that have indirect correlation with sensitive attributes, even if they are correctly identified. For example, in a case such as a lending credit operation, the *amount repayable for the target customer* (not measurable directly) is an important attribute that we want to estimate. In this case, if a measurable attribute (i.e., may be an input variable to the system) is clearly identified as discrimination cause in the estimation, such as gender, or if it is a variable that can be clearly identified as an input to the target estimation function, such as *loan amount*, the appropriate process would be rather obvious. On the other hand, for example, attributes such as *income* are closely correlated with *repayable amount* by nature, but may also reflect social disparity and discrimination. Therefore, there is a possibility that we may want to use them from the perspective of improving estimation performance, but from a fairness perspective, maybe shouldn't use it. In such a case, it is difficult to determine the strategy for ensuring fairness, and explanations are also important.

8.3.6. Attacks against AI in use

Lastly, let us note that the possible attacks against fair machine learning systems by users should be assumed. For example, in a system that performs continuous learning, an attacker may embed unfairness into the system by continuously feeding biased data, etc., or conversely, an attacker may gain by feeding hostile data to a machine learning model that is distorted by the process of making it fair.

8.4. Basic Approach to Ensure Fairness

In this chapter, we summarize the approach of fairness quality management in machine

learning module that this document address, based on the issues enumerated in the previous section.

8.4.1. Structural Model for Ensuring Fairness

Generally, fairness requirements for products and services between users and service providers (quality in use) are expressed rather conceptually and only qualitative requirements such as *not to be treated unfairly* or *to be treated equally* are described there. On the other hand, *fairness metrics* (Section 8.5.2.4.2.1), which are dealt with in the literature on machine learning AI techniques, quantify the degree of bias (bias) of various input data sets and output results focusing on certain characteristics in the stages of system development & operation. They are numerical index corresponding to the *internal quality index* in this guideline.

Based on this perspective, this guideline assumes the following process as a way to ensure fairness in machine learning systems.

- Handle qualitatively, high level requirements for fairness (usually *equal treatment* type), such as derived from clear social demands or found through *quality in use* discussion.
- Take the risk analysis based approach, which regards the compromise of equal treatment as a risk, to detail the fairness issue.
- As the work progresses, define quantitative fairness metrics on equal outcome (i.e, measurable indicator) if appropriate, and use those metrics to achieve the target.

This approach tries to ensure the appropriateness of metrics selection by means of analysis and design, similarly, to risk analysis-based approaches in functional safety area.

Figure 18 below illustrates a typical flow of this approach. This diagram is not meant to constrain individual development policies or phases, but to provide a model for shifting from qualitative considerations to quantitative methods as following.

(a) Fairness demands at the highest level of abstraction, usually demanded from perspectives such as *justice* or *human rights*, would be thought to derive from matters expressed in key words such as *equality* and *equal treatment*.
(b) From the viewpoint of social rules such as legal system and implicit ethical behavior, there are two cases where *equal treatment* is required, and where *equal outcome* are required in the form of numerical target, as shown in Section 8.3.1

In the latter case, it may mean a response assuming that the treatment is not equal in the actual institutional aspect. For example, in the context of equal employment opportunity for men and women, when there is a positive feedback structure of

undesirable(negative) phenomenon, such as that the lack of women's advancement in society hindering the development of women with sufficient social experience, which in turn delays women's advancement in society, a positive feedback may be induced to positive phenomenon by forcibly promoting women's advancement through numerical targets. In this case, since the achievement of numerical equality of outcomes becomes the primary objective, all subsequent stages are also attributed to the achievement of numerical equality.

In addition, as mentioned before, *equal treatment* has two levels; a strong requirement to *actively design and implement the system to achieve equal treatment*, and a weak requirement to *avoid intentionally providing unequal treatment*, which is a non-binding, just an effort target. This document mainly deals with the positive activities of the former and does not cover weak requirements at the level of mere effort targets.

(c) Regarding quality in use of the overall system design, and

(d) the external quality, which corresponds to the design of machine learning modules, there are two options in goal setting: numerical(measurable) equality of outcome and equality of treatment. At this stage, if the sensitive attributes to be considered are clear, and if we can envision how fairness with respect to those attributes will manifest itself in the results of the machine learning model, we can define fairness metrics, i.e., quantitative goals, that can be used in later steps.

(e) Next, in the preliminary preparation stage, where we construct(develop) the machine learning system that we want to train and consider some of the internal qualities (A-1 and A-2 as defined in this guideline), we have the same two options for setting goals. At this stage, analysis may be conducted based on the actual data collected and other information, and goals may be set.

(f) Finally, in the stage of learning and checking internal quality, there are three possible options: analyzing the statistical distribution of results, monitoring statistical and analytical indicators other than the distribution of results, and explaining the equal treatment from the logical structure of the development.

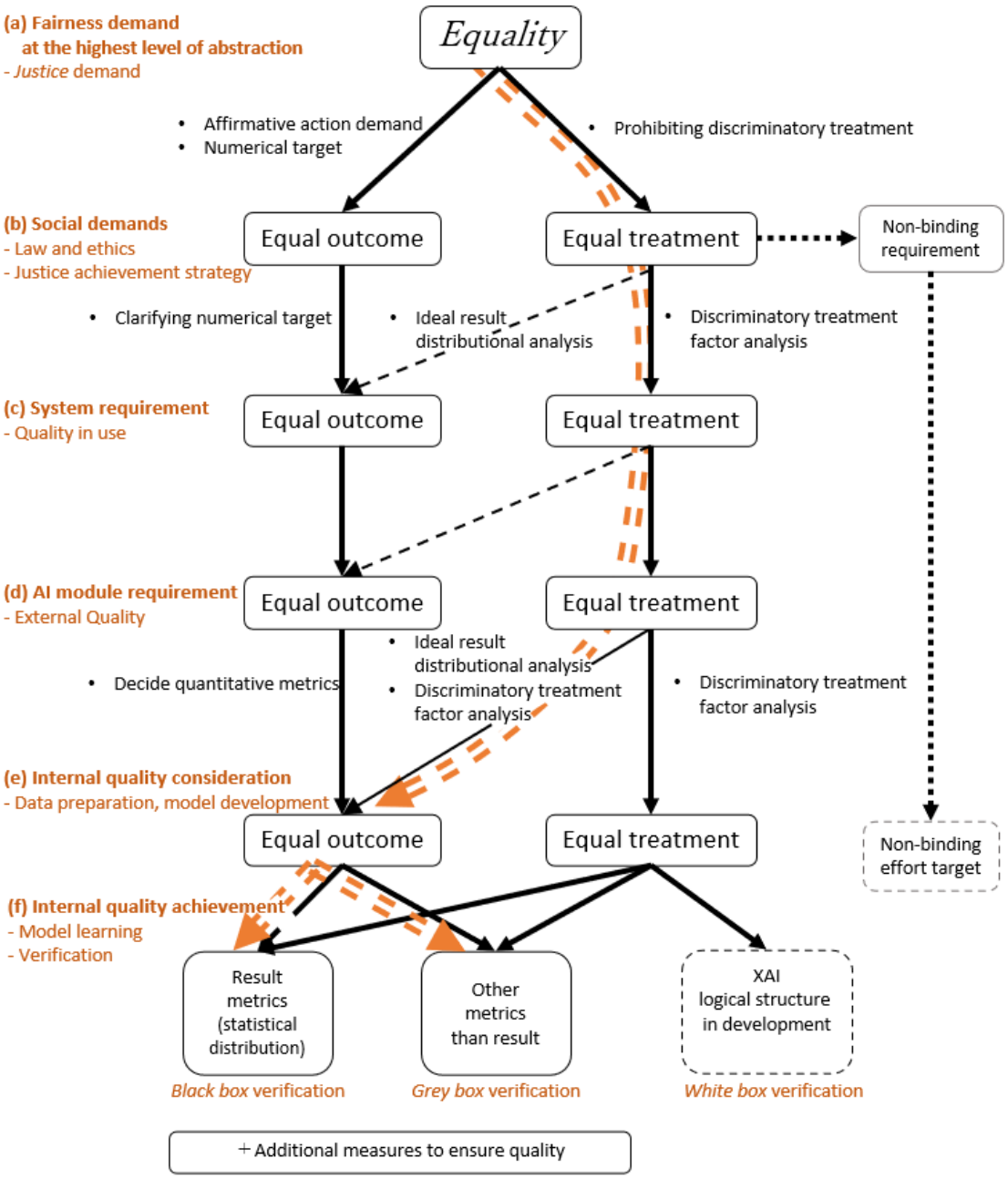

Figure 18: Example of a process structure for ensuring fairness quality

8.4.2. Basic ideas of the approach

In the remainder of this chapter, we assume the process shown by the dotted line in Figure 18 approach that can be generally considered in basic fairness management in machine learning modules. Firstly, at the level of external quality, we discuss the fairness to be ensured among features and attributes in accordance with the definition of fairness as shown

in Section 1.5.3 and at the stage of internal quality consideration, we analyze data distribution, etc., and convert them into numerical requirements for bias in outcome, and others such as in dataset if applicable. Then, based on the results of the previous-step analysis, the requirement for bias in outcome (i.e., inequity regarding distribution of results) will be measured at the system development and testing stages, together with other metrics if defined, thus confirming fairness we are aiming for.

This is based on the following two considerations: (Point 1) the fairness requirements should be reflected in the distributions of training and test datasets, since machine learning implementations are derived from data, and (Point 2) in order to sort out the complex problem structure discussed in Section 8.3, consideration based on actual data at model development stage is necessary , thus keeping the original abstract-level fairness requirements until that stage is important..

Of course, the development processes shown in the figure can be selected according to the situation of the functional requirements of the individual system. For example, in the case of an application where numerical targets are set in advance at stage (b), the implementation will be based on the functional requirements to achieve the numerical targets.

Note that the *avoidance of intentional unfair treatment as a non-binding target* described in the previous section is mapped to a universal goal for product services at the AIFL 0 level. (it is not equivalent to *best effort* as required with AIFL 1)

8.4.3. Considerations for handling data with sensitive attributes

Among personal information, there are data various sensitive attribute, such as race, gender, place of origin, etc., that require careful handling and must avoid unfair treatment resulting from those values. In developing machine learning modules that require fairness, the careful handling of these sensitive attributes (data) is required from the early stages of the process shown in Figure 18.

This document does not take the position that it becomes automatically fair if it does not handle these sensitive data. Data obtained from the real world have complex structures and correlations, and it is quite possible that AI built from other input data than sensitive data will eventually be found to have correlations with the sensitive data which is even not included in the input. Furthermore, not using the sensitive attributes in the development (especially in the testing process) can be a major disadvantage because it makes it impossible to check and inspect the quality of fairness. This is because sensitive attributes are often the key points for ensuring fairness. Therefore, in order to build a fair machine learning system, we believe that such sensitive data should at least be obtained during development, and sufficient consideration should be given at the stage of system design and development process consideration.

On the other hand, whether or not sensitive data can be obtained during operation phase

for the purpose of quality monitoring and additional learning, needs to be considered in relation to personal information protection. In addition, there may be cases where it is impossible to handle sensitive information that requires special consideration, such as race or medical history, even at the system development stage. In these cases, it is necessary to give sufficient consideration to how to eliminate bias and conduct monitoring in the model building and additional learning stages.

8.5. Quality Management for Fairness

8.5.1. Refinement of fairness requirements as preliminary preparation

If the required fairness target level is assumed to be AIFL 1 or higher, the points described below shall be considered in the requirements definition stage. If multiple factors are related to fairness, the level shall be determined in principle, corresponding to the most demanding factor.

8.5.1.1. Clarification of fairness goals

First of all, the goal of ensuring fairness for the target system should be clarified as follows.

(1) Consider the attributes that form the basis for judgments about fairness.

A) Clarify the information that needs to be taken into consideration for fairness, (i.e., *sensitive attributes*) from the target function's point of view. This information is not limited to that which is explicitly identified as an attribute of the actual input data. For example, information such as gender, age, race, and ancestry could be identified as sensitive attributes even though not included in data.

B) If possible, clarify the observable information (e.g., written test scores, department of choice, etc.) that may be used for judgment with sufficient consideration of the influence from the sensitive attributes clarified above.

C) Furthermore, if possible, verbalize the essential characteristics (potential academic ability, graduation potential, etc.) that are the source of the observable attributes to be judged above and are not affected by the sensitive attributes.

Here, it should be noted that the results of the above study and subsequent efforts will vary

depending on the scope of fairness that the target system (or social activity that includes it) should consider, or the assumptions of *indicators to be considered fair/unfair* given in the pre-condition.

(Example.)

For example, consider a college admission examination system, and the scoring and pass/fail judgment process of the system is our target of discussion.

From the standpoint that solving the same prepared test questions at the same time shall give examinees a fair chance, the *validity of the answers to the test questions* becomes an essential (abstract level) index that should be used for the final judgment of pass/fail, and whether the *score result* might be an appropriate observable numerical index for it.

On the other hand, as in the case of college entrance exams discussed in the fairness related study [52], the examination system itself may have fairness problems. From their point of view, *potential academic ability* and *possibility of graduation* may be essential ideal criteria for the judgement. In addition, the *score result* would be affected by the disadvantages derived from the poor past educational opportunities, might be strongly correlated racial, income data. They may argue that some corrections considering those aspects might be necessary for a fair pass/fail decision close to the ideal criteria.

(2) Clarify the basic handling policy for the sensitive data.

A) Does the sensitive data require that the system not be treated in a discriminatory manner because of the information (equal treatment), or does it require that the results should follow a certain predefined distribution with the specific numerical targets (equal outcome)?

Are there any legal requirements regarding the handling of the information in question, such as those that exist in the areas of equal employment opportunity, equal race, etc.?

B) If the discrimination related to the sensitive data already exists in the real world, how should the AI, which is built based on the data collected from the real world, handle it? Should such discrimination be removed as much as possible, or can a certain amount of residual discrimination be tolerated, or should be reproduced also within AI output based on the system requirements?

C) Clarify the constraints on the available measures to remove residual discriminatory judgments in machine learning results. Is it permissible to introduce intentional

corrections to the training data or to intentionally adjust the trained model? It should also be clarified how the decision to stop development will be made if the problem cannot be corrected by the available means.

After making the above decisions, the fairness requirement level (AIFL) is checked again.

8.5.1.2. Modeling dependencies and causal relationships among data attributes

Since data acquired from the real world has complex structures and correlations, machine learning without properly understanding the dependencies and causal relationships between attributes may cause inappropriate for the purpose or inaccurate results.

As we plan to implement the system using machine learning with data rather than logical description by program description, the dependencies and causal relationships among attributes are not completely known (if they were known, the non-AI program could have been written), and it is difficult to grasp them completely even after exhaustive analysis. However, it is very important to investigate especially the following aspects as far in advance as possible.

1) Information paths that may cause unfairness

In the process by which a sensitive attribute unfairly affects the output of a system, the sensitive attribute may directly affect the result, or there may also be contributions from other attributes that are affected by the sensitive attribute, i.e., indirect influence trough those attributes. Without understanding these series of causal *paths*, it is often the case that the final fairness requirement cannot be met nor explained, even removing the direct impact.

In addition, when there are attributes correlated both the attributes; that should contribute to the system decision and that should not, a typical pre-processing such as eliminating bias in the dataset regarding the sensitive attribute can have a negative impact on the AI system decision performance, thus requiring careful consideration.

2) Simpson’s Paradox.

Simpson's paradox refers to the phenomenon that when analyzing to find a relationship between two data attributes ignoring the third factor that is actually involved in both of them, a correlation that does not exist in reality or a correlation that is the opposite of reality may be shown [100]. This paradox was proposed over 50 years ago in the field of statistics, but it also applies to machine learning today. In order to prevent such a phenomenon, it is necessary

to understand the relevant third attribute in advance and to prepare learning data with *case separation* for the third attribute. We will discuss specific examples later.

3) Necessity of using sensitive attributes

There can be cases where fairness cannot be achieved without daring to use sensitive attributes. In other words, fairness by unawareness may sometimes be insufficient for the objective.

For the support of analysis described above, a graphical method that can represent these interrelationships in a relatively concise manner, called *Causal Bayesian Networks (CBNs)* [52] will be presented here. CBNs are graphical representations that exhibits the causal path between attributes, helping us to consider the patterns of unfairness underlying the training data. It is also useful for examining the types of fairness metrics that should be used.

(Example.)

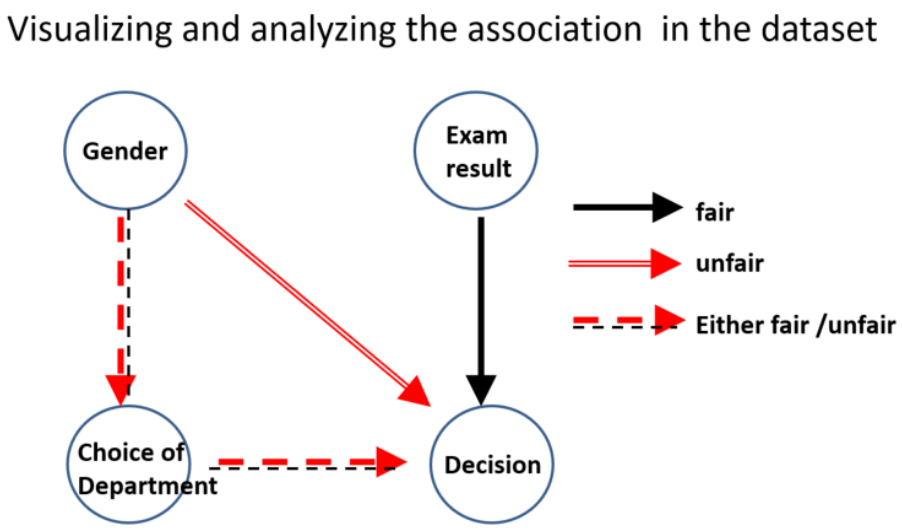

Figure 19: Example of association visualization using CBN (from [52])

Figure 19 shows a college admission scenario in which applicants are admitted based on qualification exam result, choice of department, and gender discussed in [52], etc. CBN visualizes the relationship between attributes by representing a direct influence of α on β is represented by an arrow from α → β (causal path), which is to be considered to see whether their influence is fair or unfair, or “either is possible”.

In this example, the causal path from exam results to pass/fail decision is certainly fair. If gender is used directly in the pass/fail decision, i.e., if the two individuals with same exam results can be treated differently just depending on their gender, there exist the causal path gender → decision, which is certainly unfair.

The path from gender to choice of department could be either as fair or as unfair. If the fact that women tend to prefer certain department is due to implicit environmental pressures on women, then it is unfair causal path. Similarly, the path from choice of department to decision could either be considered as fair or as unfair. If the admission rates for departments chosen more often by women are intentionally lowered, then the path would be unfair.

Thus, by carefully analyzing the paths that may influence pass/fail decisions, we can see that the path gender- choice of department- decision can be a factor that impairs fairness, depending on the actual situation and algorithm. This means that the fairness of the decision result cannot be guaranteed by simply excluding the direct causal path from gender to decision.

Figure 20 is an example of the Simpson's Paradox in CBNs. For the purpose of finding correlation between the attribute exercise days per week and the attribute health (assuming, some health parameter), we may not be able to achieve our goal if we train our model using dataset collected without considering the third factor age, which affects both attributes.

For simplicity's sake, let's assume that age is positively correlated with exercise day per week and negatively correlated with health. In this case, even if exercise and health are positively correlated actually, the opposite result may occur. The third factor, such as age in this case, is referred to as confounding factors and should be carefully handled in training dataset preparation, not only from a fairness perspective. Drawing CBNs encourages the discovery of such confounders, and if an attribute of concern is a confounder, it can be recognized in advance that simply removing it may not be appropriate.

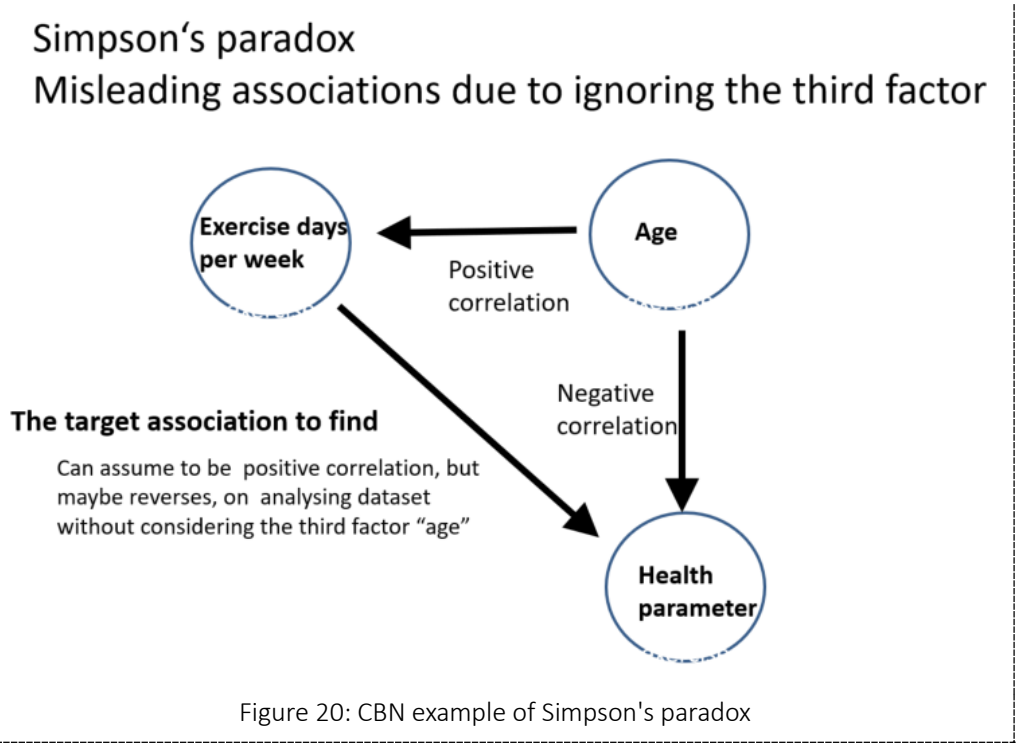

Figure 20: CBN example of Simpson's paradox

8.5.2. Fulfillment of fairness requirements

8.5.2.1. Policy Outline

This section provides an overview of the actions for meeting the fairness requirements. Depending on the scope of the techniques, they can be broadly categorized as follows.

A) Changing/Adjusting training datasets
B) Techniques on learning algorithm
C) Adjustment to outcome (trained models)
D) Correction in use

These also correspond to the *three categories of fairness techniques by development process* [87][134][98] as follows.

A): Pre-processing approach

B): in-processing approach

C) and D): Post-processing approach

Depending on developers’ ability to intervene in the process of the target machine learning module, the choice among measures would be made as follows.

– If the developer can modify training dataset, then pre-processing is worth trying because it allows for relatively straightforward explanations such as "we trained on a dataset with specific metrics and guaranteed fairness" and is generally quite effective.

– If the developer can design model/learning algorithm, then in-processing approach can be used.

– If the develop can only receive a trained model (basically as back box) and can perform some adjustment to it, then only post-processing would be the choice.

Taking into account the target level of fairness, the basic policy of actions is approximately as shown in the following table.

Table 5: Summary of work scope and actions by AIFL

	When the training dataset can be modified	When the design model and learning algorithm can be modified	When only post-processing of trained models is possible
AIFL 2	Define and record fairness metrics *goals* for the training dataset, with multiple pre-processing techniques, if necessary, to improve and ensure they are met as much as possible.	Add fairness metrics for model output to the learning objectives, and balance them with other metrics such as AI performance, and to achieve the target using in-processing techniques.	N/A (Consider in advance, the measures that can be taken to adjust the trained model. If the essential conditions are unlikely to be met, the work scope needs to be reviewed.)
AIFL 1	Define and measure fairness metrics *goals* for the training dataset. If there are deviations, pre-processing techniques are used to improve and recorded with the result.	Fairness metrics for model output are added to the learning goals, and if the results of training with AI performance priority deviate from the goals, improve it by in-processing techniques and recorded with the result.	Define fairness metrics for model output and measure them during testing. If there is a deviation from the target, try to improve it by adjusting the trained model, or any other adjustment in the operation.
AIFL 0	Define, measure and record fairness metrics for training datasets.	Define fairness metrics for model outputs and record their measurement during training.	Define fairness metrics for model outputs and record the measurements during testing.

The following sections describe the details of the techniques, mainly in line with the internal quality set forth in Section 1.7 and Chapter 6.

8.5.2.2. A-1: Sufficiency of problem domain analysis (Preliminary Preparation)

By following the guide presented in the preliminary preparation (8.5.1), the following aspects can be investigated.

– Dependency analysis between attributes

– Clarification of requirements for training dataset distribution

– Metrics to be checked (fairness metrics)

In addition, considering the nature of the overall system to be developed, the requirements of the following perspectives should be sufficiently confirmed.

– Consistency with legal and social requirements

Since the development team may not have sufficient knowledge of legal and social requirements, it is advisable to consult experts in the field.

The required actions for each target AIFL level are as follows in the preliminary preparation phase.

- AIFL 1
  - Define the fairness requirements and record them, including the history.
  - Based on this requirement, requirements for data preparation (fairness metrics for training datasets) shall be defined.
  - Based on this requirement, fairness metrics for model outputs shall be defined.
- AIFL 2
  - In addition to those listed in AIFL 1, the following actions shall be taken
  - To model dependencies and causal relationships between data attributes.
  - The results of modeling should be reflected in the fairness metrics of the training data set.

The *fairness metrics* for the training dataset are basically from the same perspective as those for the model output. For example, if the fairness requirement is *no difference in pass/fail judgments based on gender* for the output, the training dataset should also be free of bias from that perspective. Examples of metrics are described in 8.5.2.4.2.1

8.5.2.3. B-1 to 3: Coverage/Uniformity/Validity of datasets (Data Preparation)

In the data preparation stage, *pre-processing* techniques are taken to ensure required fairness,

mainly from the internal quality perspective of *coverage* , *uniformity*, and *validity* of datasets. What is commonly referred to as Data Fairness is achieved at this stage, and there are two types to address it; one for data collection process and the other for the data collected.

8.5.2.3.1. Data collection process without introducing bias

From the perspective of preventing *disparate treatment*, first of all, we must avoid bias in data collection operation. such as sample size disparity and tainted examples[11], i.e., datasets become biased against the real world.

Actions here are based on the perspective of *coverage* and *uniformity* of datasets in the sense of confirming whether there is enough data for the attribute combinations of interest, or whether the data is close to the distribution in the real world. Techniques such as *weighting the collected data by groups* (Reweighing [76]) are used for the purpose.

8.5.2.3.2. Data adjustment, enhancement, and synthesis

If we want to prevent *disparate impact*, even if the dataset corrected is not biased against reality, we may need to take measures against the distortions and discriminatory factors inherent in the dataset from the real world, corresponding to *validity* of datasets point of view.

Examples of major techniques include the following

- Disparate Impact Remover

  To remove the bias caused by the so-called "proxy", the attribute value that may be correlated to the sensitive attribute is modified to reduce the possibility of indirectly inferring the sensitive attribute from that proxy attribute.

- Optimized Pre-Processing

  A framework for pipelined transformations to pre-process training data. Consideration is given to both the removal of bias and the usefulness of the data, i.e., not too far from the real data [47].

8.5.2.3.3. Required actions for each quality level in data preparation

Required actions for each target AIFL level are as follows in dataset preparation phase.

[11] Sample size disparity: The number of data varies greatly depending on the value of the attribute requiring consideration.
Tainted Sample: Bias caused by human labeling

- AIFL 1
  - Measure and record fairness metrics for datasets.
  - When the dataset preparation is within the work scope, if the metrics deviate from the target, at least *different treatment* measures described in 8.5.2.3.1 shall be taken to improve the metrics.

    –
- AIFL 2
  - In addition to those listed in AIFL 1, the following actions shall be taken
  - When the dataset preparation is within the work scope, if the metrics deviate from the target, *different impact* measures described in 8.5.2.3.2 shall be taken.
  - When dataset preparation is outside the work scope, consider measures to minimize the impact of the dataset metrics deviation from the target on learned model output later in the process. In some cases, collateral outside of the machine learning component should be considered in advance.

(Note) Some of the pre-processing methods described herein are available on a development platform and tool as described in Section 8.6. However, in almost call the cases there, the input data is assumed to be a series of *attributes and values* (structured data) and *sensitive attributes* are already identified within the data. it is sometimes difficult to use it directly for the target machine learning system. For example, natural language processing that takes a large amount of text (corpus) as input. Fairness in natural language processing, Mehrabi et al. [87] have introduced bias removal methods in language modeling and feature learning methods, but further research is still needed to make these methods widely available in the field. At present, a realistic approach is to ensure fairness by testing and adjusting the resulting machine learning models in post-processing.

8.5.2.4. C-1 to 2: Model accuracy and stability (Training and Testing)

In training with the dataset, *in-processing* and *post-processing* techniques are taken to ensure fairness from the internal quality *accuracy of the model* and *stability of the model* perspectives.

8.5.2.4.1. Countermeasures (in-processing) in model development and learning phase

In-processing measures are those that aim to embed fairness in the model and can mainly be implemented by modifying the objective function (adding constraints) or adding model elements, as described below.

8.5.2.4.1.1. Example of additional constraint technique

– Prejudice Remover

The resultant probability distribution is adjusted by adding a constraint to remove indirect prejudice[12] that remains even if the sensitive attributes are not used directly. For this purpose, the constraints are implemented as a regularizer using the sensitive attributes and added to the objective function.

This method is highly explanatory, although its applicability is limited to identification problems [72].

8.5.2.4.1.2. Example of adding model elements technique

– Adversarial Debiasing

This technique adds an adversarial model that uses the inference results from the target model as input to infer the sensitive attribute. The target model should be trained so that the inference of that adversarial model fails as much as possible.

Adversarial debiasing allows for a wide variety of target model algorithm and may handle the problem of insufficient proxy analysis in data preparation phase. [125].

8.5.2.4.2. Adjustment to trained models

Among the measures classified as post-processing techniques, this section discusses *adjustments to the trained model*, from model accuracy and model stability aspects, while in the next section, *adjustments during operation assuming that discrimination remains* will be discussed.

Depending on the required fairness goal metrics for the trained model, there are adjustments such as the Equalized Odds Framework and calibration [102].

Of course, quantitative evaluation of target metrics is necessary during learning as an internal quality target, and it is essential to define and measure them appropriately.

[12] One source of bias. The statistical dependence of a target variable or a non-sensitive attribute on a sensitive attribute.

8.5.2.4.2.1. Example of target metrics

All of the metrics are positioned as indicators to confirm that there is no *different impact* discrimination. They are selected according to the concept of discrimination concerned for the system. Some commonly used fairness metrics are listed below. For others, please refer to the literature [87].

Equalized odds	Make the *right decision* without relying on a sensitive attribute. ($TP/(TP+FN)$ and $TN/(FP+TN)$) are equal.
Predictive parity	Regardless of a sensitive attribute, the *precision rate* ($TP/(TP+FP)$) are equal.
Demographic parity	The percentage of *Desired outcome* is equal regardless of a sensitive attribute. $(TP+FP)/(TP+FP+TN+FN)$ Desired outcome (equal percentage)

8.5.2.4.3. Required actions for each quality level in model training

Required actions for each target AIFL level are as follows in model training phase.

- AIFL 1
  - Measure and record the model output metrics.
  - If there is a deviation from the target, at least one in/post-processing technique shall be used to make improvements, and if there is still a deviation, record it.
- AIFL 2
  - In addition to those listed in AIFL 1, the following actions shall be taken
  - If there is a deviation from the target, try multiple in/post methods to make improvement as much as possible.
  - If there is still a discrepancy that cannot be resolved, consider combining additional measures in actual operation, sometimes extending to outside the machine learning module.

8.5.2.5. Adjustment and usage during operation

Even with the measures described so far, the bias may not be fully removed. To deal with such

cases, there are measures to adjust the learned model in some way when actually performing inference. The following is one example of such methods that has been proposed. Of course, other adjustments can be made to the system as a whole, including outside the machine learning elements, to suit the situation.

– Reject Option-based Classification (ROC)

  Based on the hypothesis that discrimination occurs when the *confidence level* of the model inference is low, ROC rejects or revises the results of inferences that may lead to discrimination. Specifically, if the *favorable outcome* of the *favored group* or the *unfavorable outcome* of the *non-favored group* comes out lower than the specified confidence level, the result is rewritten [75].

8.5.2.6. Maintainability of quality in operation (E-1)

Products and services for which fairness is important are often used in public spaces and other open environments, and therefore, sufficient consideration must be given to changes in quality during operation, such as concept drift. It is necessary to measure the fairness metrics for model outputs and datasets at appropriate intervals, regardless of the target AIFL level.

8.6. Development infrastructure and tools for fairness

8.6.1. Purpose of using development platforms and tools

The work described in Section 8.5performed in the machine learning lifecycle, along with other quality goals than fairness. Fairness often requires a trial-and-error approach, including the identification of biases inherent in datasets and the measurement of fairness metrics for trained models. For this reason, the use of development platforms and tools is quite beneficial for the followings:

– Efficiently consider the perspectives described in Section 8.5.2 for the dataset.
– Perform some kind of visualization, metrics evaluation, or other fairness assessment of the trained model.
– Maintain appropriate records of work, the datasets used in the process, and the sequence of work.

8.6.2. Examples for applicable tools

8.6.2.1. Typical functions and modules

The main functional groups that are desired for the platforms/tools are listed below, based on the purpose described in the previous section.

(a) Visualization of datasets (analysis to discover relationships regarding sensitive attributes)
(b) Discovering fairness matters other than those clarified with metrics (e.g., counterfactual experiments to test the impact of changes in certain attribute values on output decisions)
(c) Various fairness metrics measurement
(d) XAI Library (visualization of which attributes contributed to the decision)
(e) Infrastructure for building pipelines to support the lifecycle of continuous monitoring and relearning after operation (so-called DevOps infrastructure + infrastructure for data sets and models)

Google and IBM Fairness360 are described below for reference, but new features from various other vendors will continue to be provided, so please check the latest information before your selecting the appropriate one.

8.6.2.2. Example 1: Google tools

Functions (a) through (e) described in the previous section will be covered as follows.

- What-if-tool (a), (b)
  - https://pair-code.github.io/what-if-tool/learn/tutorials/walkthrough/
- Fairness Indicators (c)
- Explainable AI (d)
  - https://cloud.google.com/explainable-ai/
- AI-Platform (e)
  - https://cloud.google.com/ai-platform

The What-If-tool, which provides a variety of functions directly related to fairness, is not just a visualization of the training dataset, but also helps to discover potential bias patterns in the dataset through hypothetical scenarios and analysis functions when sliced by various attributes.

Explainable AI can also contribute to confirming the bias, as it can quantitatively indicate which attributes contributed to the decision during inference. (Note that, as mentioned in 8.2having influence does not necessarily mean unfairness.)

8.6.2.3. Example 2: IBM tools

Functions (a) through (e) described in the previous section will be covered as follows.

- AI Fairness 360 / IBM Watson OpenScale … (a), (b), (c)
  - https://aif360.mybluemix.net/
  - https://www.ibm.com/jp-ja/cloud/watson-openscale
- AI Explainability 360 / IBM Watson OpenScale … (d)
  - https://aix360.mybluemix.net/
  - https://www.ibm.com/jp-ja/products/cloud-pak-for-data
- IBM Cloud Pak for Data … (e)

AI Fairness 360 and AI Explainability 360 are open source software developed and released by IBM Research Foundation and donated to the Linux Foundation in 2020. AI Fairness 360 provides a rich set of features to address biases in machine learning models and datasets, and generally covers the methodologies described in this document.

- Pre-processing: Supports reweighing of training data, disparate impact remover, optimized preprocessing of training data, etc.
- In-processing: supports prejudice remover, adversarial debiasing, etc.
- Post-Processing: Equalized odds and calibrated equalized odds are supported.
- Metrics: Predictive parity and others are supported; Equalized odds and demographic parity can be calculated using the provided modules.

IBM Cloud Pak for Data is a multi-cloud software that covers the lifecycle of machine learning models in an integrated manner, and IBM Watson OpenScale is one of the software modules that make it up. The ability to monitor input and output data during model operation and detect time-series changes in fairness metrics is useful for improving biases that occur after operation begins.

9. Security

9.1. General Principles

In this chapter, we summarize the main issues in the security of machine learning based systems and machine learning components.

The basic approach to the security of machine learning based systems is analogous to that of conventional information systems; when constructing a machine learning based system, we analyze possible threats and vulnerabilities by using *attack trees* and take countermeasures against them. In the threat analyses, we investigate not only the attack surfaces, threats, and vulnerabilities found in conventional information systems, but also those unique to machine learning technologies, which require to take additional countermeasures. In Section 9.2, we describe well-known attack surfaces, threats, and vulnerabilities in machine learning based systems. In Section 9.3, we show basic ideas and techniques for the countermeasures against attacks as reference.

The analyses and countermeasures presented in this chapter do not identify all possible threats and vulnerabilities or counter all possible attacks; the attacks listed in this chapter should be regarded just as examples. The developers of machine learning based systems need to analyze the attack surfaces, threats, and vulnerabilities in the design, development, operation, and maintenance of the systems. In particular, they should collect the latest information on security issues when analyzing the threats and vulnerabilities. Furthermore, they also need to perform these analyses in the periodic security risk assessments.

Finally, we mention a few related documents as references. NIST IR 8269 [31], a draft on the taxonomy and terminology of adversarial machine learning, discusses the security of machine learning based systems from the three perspectives: *attack*, *defense*, and *impact*. NIST SP 800-30 [32] is a guideline for risk assessments of information systems in general.

9.2. Classification of attacks on machine learning based systems

In this section, we introduce four categories of attacks on machine learning based systems (Section 9.2.1) and present examples of attack methods (Section 9.2.2).

9.2.1. Categories of attacks on machine learning based systems

In this guideline, we classify attacks on machine learning based systems (hereafter referred to as *systems*) into the following four categories:

A) Malfunctions of the trained model and systems (Section 9.2.1.1)

B) Leakage of information on the trained model (Section 9.2.1.2)

C) Leakage of sensitive information on the training data (Section 9.2.1.3)

D) Malfunctions in the interpretation/explanation functionalities of models and systems (Section 9.2.1.4)

9.2.1.1. Malfunction of models and systems

Attacks on trained models and systems can cause conflicts with the functional requirements of the machine learning based systems. Here are examples of the possible problems caused by such attacks:

– Reduction of safety/increase in risks:

  - Accidents by evading object detection in autonomous driving, and by missing abnormalities in the driver;
  - Evasion of malware detection in computer systems;
  - Evasion of intrusion detection, and failure of the detection of abnormal behaviors and dubious persons in security systems;
  - Increase in false positives and false negatives in pathological diagnosis.

– Decrease in AI performance:

  - Failures in face recognition and other authentication systems;
  - Traffic congestion, increase in logistics costs, and inefficient allocation of vehicles in transportation/logistics;
  - Decrease in the accuracy of product recommendation and demand prediction in the retail sector;
  - Decrease in profits from stock trading, losses due to inappropriate financing;
  - Inadequate enrollment, hiring, and staffing.

– Decrease in AI fairness:

- Discriminatory lending through credit screening systems;
- Unfair admissions and employment;
- Discriminatory crime risk assessments by security systems;
- Inequitable treatment effects caused by pathological diagnostic systems.

There are (i) attack methods causing a malfunction of a specific model and (ii) those resulting in malfunctions of various models with high probabilities.

9.2.1.2. Leakage of information on models

An attack that leaks information about a trained model (e.g., parameters and hyperparameters of the model) may violate the intellectual property right of the trained model. Furthermore, this attack can be used to mount other kinds of attacks on security and privacy:

– Information on a trained model may be used to generate inputs that cause the malfunctions of the system.

– Information on the parameters of a trained model may leak some sensitive information in the training data used to train the model.

9.2.1.3. Leakage of sensitive information on the training data

Sensitive information about the training data used to train the model may be obtained by observing the trained model's behavior. In particular, if the training dataset contains sensitive personal data, we may need to take countermeasures against this kind of attack.

9.2.1.4. Malfunctions in the interpretation/explanation functionalities

If a trained model or a system has a functionality to provide an interpretation/explanation of its behavior, then an attack on this functionality may result in the degradation of the quality-in-use, transparency, and accountability of the system.

9.2.2. Examples of attack methods

Attack methods are classified into three categories in terms of the adversary's knowledge: (a) *white-box attacks* (attacks that use the parameters of the trained model and possibly other

information of the system), (b) *black-box attacks* (attacks that do not use the parameters of the trained model or the system), and (c) *gray-box attacks* (between (a) and (b)). In black-box attacks, the adversary typically uses information obtained by querying input data to a trained model or system and observing the output of the model or system.

In the rest of this subsection, we present examples of attack methods: (i) those for conventional information systems (Section 9.2.2.1) and (ii) those specific to machine learning (Section 9.2.2.2).

9.2.2.1. Attacks in terms of conventional information security

9.2.2.1.1. Attacks on the data collection process

An attack on the data collection process can result in collecting data that violate security, privacy, intellectual property, legal regulations, and contracts. Then the trained model may learn sensitive information in the collected data. For example, the training of models may use personal information collected without the consent of individuals or confidential image data of military facilities. This attack may result in disclosing sensitive information that violates legal regulations or contracts when combined with another attack to leak information about the training data (Section 9.2.2.2.5).

Furthermore, an attack on the data collection process may manipulate the training dataset to trigger a malfunction of a trained model (a data poisoning attack, discussed in Section 9.2.2.2.1).

9.2.2.1.2. Leakage of hyperparameters and datasets

The hyperparameters and the training datasets used in training may be used to mount other attacks against the system, for example, to generate adversarial inputs. Therefore, it may be necessary not to disclose or leak the hyperparameters and the training datasets. This may be helpful to prevent/mitigate white-box attacks unique to machine learning.

9.2.2.1.3. Vulnerabilities of machine learning framework software

The machine learning framework software's vulnerabilities may allow for the adversary to embed malware and backdoors in the machine learning framework itself or in pre-trained models. These malware or backdoors may be used to mount attacks on the trained model or system.

9.2.2.1.4. Manipulation of test data

If the reliability of the test dataset is not confirmed appropriately, some manipulated test data may be used to evaluate a trained model. This would cause the failure of detecting a malfunction of the trained model for specific input data, hence overlooking attacks on the

system.

9.2.2.1.5. Leakage of trained models

The parameters of a trained model can be used to mount attacks, e.g., to generate adversarial inputs against the model. This kind of attack may be more efficient than the attacks using the hyperparameters and training datasets (Section 9.2.2.1.2); therefore, it is more important to prevent the leakage of the information on the trained models.

9.2.2.1.6. Manipulation of trained models and interpretation/explanation functionalities

If adversaries manipulate the trained models or their interpretation/explanation functionalities, then they can mount various attacks, such as a backdoor embedding and a malfunction of the system.

9.2.2.2. Attacks specific to machine learning

Attacks specific to machine learning can be classified according to the phases of the attacks.

- Attacks during or before the training phase: e.g., data poisoning attacks (Section 9.2.2.2.1) and model poisoning attacks (Section 9.2.2.2.2).
- Attacks during the operation phase: e.g., evasion attacks (Section 9.2.2.2.3), model extraction attacks (Section 9.2.2.2.4), data privacy attacks (Section 9.2.2.2.5), and attacks to malfunction the interpretation/explanation functionalities (Section 9.2.2.2.6).

9.2.2.2.1. Data poisoning attacks

A *data poisoning attack* is an attack during or before the training phase that aims to cause a malfunction of a trained model (e.g., to degrade the accuracy of a classifier) by injecting malicious data into the training dataset. There are two types of attacks: those that degrade the performance of a model or system for unspecified inputs during the operation phase [68], and those that degrade the performance of a model or system only for specific inputs during the operation phase (*backdoor attacks*) [49].

In backdoor attacks, a model or system malfunctions only when the input during the operation phase contains certain information that triggers the malfunction (e.g., when a specific symbol appears in the input image). Since the system does not malfunction for any other inputs than the specific one, backdoor attacks are more difficult to detect in general.

In data poisoning attacks, an injection of malicious data into the training dataset can be performed by attacking the data collection process (Section 9.2.2.1.1), including the direct

manipulation of the training dataset.

9.2.2.2.2. Model poisoning attacks

A *model poisoning attack* is an attack during or before the training phase that aims to cause a malfunction of a trained model by directly manipulating (part of) the model, e.g., gradients in a neural network. For instance, if a given pre-trained model is used for training (e.g., in differential development, transfer learning, and federated learning), then a model poisoning attack may embed malicious backdoors in the pre-trained model in advance.

9.2.2.2.3. Evasion attacks

An *evasion attack* is an attack during the operation phase that causes a malfunction of a trained model or system by providing the system with specific malicious input (called an *adversarial example*). For example, in image classification, an adversarial example used in an evasion attack is a carefully perturbed image input that appears natural to human eyes but causes misclassification of this image.

Evasion attacks can be classified into black-box attacks and white-box attacks. In white-box evasion attacks, adversarial examples are tailored to a specific model using the internal information on the machine learning algorithm and the trained model (e.g., hyperparameters and parameters). In black-box evasion attacks, adversarial examples are generated based on the black-box behavior of the trained model or system. Typically, black-box attacks rely on the transferability of adversarial examples across different training datasets and different trained models with possibly different architectures. Specifically, the adversary provides the black-box system with the adversarial examples that are generated using some different (white-box) models trained by the adversary using possibly different training datasets.

9.2.2.2.4. Model extraction attacks

A *model extraction attack* is an attack during the operation phase that aims to steal information on (i) model attributes (e.g., parameters, hyperparameters, architectures, decision boundaries) and (ii) model functionality (e.g., building a substitute model that behaves similarly to the original model) [115][70]. This kind of attack can be performed during the operation phase by observing the behavior of the output given input data. Model extraction attacks may be helpful for the adversary to mount other types of further attacks.

9.2.2.2.5. Data privacy attacks: membership inference attacks, model inversion attacks, and property inference attacks

A *data privacy attack* is an attack during the operation phase that aims to steal information on the training data used to train the model. This kind of attack may result in the leakage of sensitive information in the training data. We mention the following three categories of

attacks:

- A *membership inference attack* [108] is a data privacy attack to steal membership information on training data (i.e., whether the training dataset includes a particular input) by white-box or black-box access to the trained model during the operation phase.

- A *model inversion attack* [61] is a data privacy attack to reconstruct sensitive features of training data by white-box or black-box access to the trained model during the operation phase.

- A *property inference attack* [42] is a data privacy attack to infer a statistical property of the training dataset by white-box or black-box access to the trained model during the operation phase.

9.2.2.2.6. Attacks to malfunction the interpretation/explanation functionalities

Techniques for interpretation/explainability functionalities can malfunction for adversarial input data, since they are often black-boxes. For example, some attacks reduce the quality of the generated explanation, and some generate incorrect explanations [55][110].

9.3. Approaches to security measures

In this section, we present approaches to security measures of the following categories: (i) countermeasures at the system level (Section 9.3.1), (ii) countermeasures in the entire development process (Section 9.3.2), (iii) countermeasure for conventional information security (Section 9.3.3), and (iv) countermeasures against attacks specific to machine learning (Section 9.3.4). We often need to integrate these different categories of measures.

9.3.1. Countermeasures at the system level

- The system should be designed to reduce the opportunity for an adversary to input malicious data into the trained models and the system during the operation phase.

  - When input data are obtained from a third party or a public space (e.g., the Internet) during the operation phase, the system should confirm the reliability of the data provider and the authenticity of the input data.

  - When input data are obtained from the external environment (e.g., camera images) during the operation phase, the opportunities for attacks in the external environment should be reduced. For example, the system or the operator should limit the amount and frequency of data that a user can input into the system per unit time.

- The system or the operator may apply methods for attack detection against the trained models or the system.

- The system and the stakeholders should reduce the amount of information available for attacks.

  - The system and the stakeholders should disclose as little information as possible about the trained models and systems (e.g., hyperparameters and architectures) to reduce the adversary's prior knowledge of the system (especially to prevent white-box attacks).

  - The system should hide or perturb unnecessary information in the output (e.g., confidence scores) of the trained models and the system as much as possible to prevent or mitigate attacks.

  - The system should prevent or disturb the adversary's observation of the system's behavior to reduce the probability of successful attacks and to detect attacks more easily.

These measures do not guarantee to prevent the attacks even though they may reduce the opportunities/probabilities of successful attacks. Furthermore, note that these security measures at the system level may deteriorate the transparency and accountability of the system.

9.3.2. Countermeasures in the entire development process

- The developer should check whether the training data and pre-trained models are authentic and their providers are reliable. For example, the developer should verify digital signatures or use frameworks for providing reliable datasets to ensure that the datasets and pre-trained models have not been manipulated maliciously.

– The developer should obtain the latest information on the vulnerabilities of the software used in the system (e.g., those of the machine learning framework).

– The developer should take security measures for the development of conventional information systems that are not limited to machine learning.

9.3.3. Countermeasures for conventional information systems

The developer should take measures for the security risks of conventional information systems by referring to frameworks, such as ISO 27000 series [10], ISO/IEC 15408 (Common Criteria) [3], NIST SP800 series [32], NIST Cyber Security Framework [33], and guidelines issued by IPA (Information-technology Processing Agency of Japan).

For business perspectives, ISACs (Information Sharing and Analysis Centers) provide guidelines and references for each business field. For example, the following ISACs have been established in Japan:

– Japan automotive ISAC

– ICT ISAC Japan (https://www.ict-isac.jp/)

– Japan Electricity ISAC (https://www.je-isac.jp/)

– Financials ISAC Japan (http://www.f-isac.jp)

For industries where ISACs have not been established, it is recommended to refer to security standards for the critical infrastructure or finance area, such as PCI-DSS (Payment Card Industry Data Security Standard)[37].

9.3.4. Countermeasures against attacks specific to machine learning

At present, there are no established methods for the security risk analysis or countermeasures of machine learning based systems, although more and more researches have been conducted recently. Furthermore, there are no practical security frameworks that can be used as references for designing and developing machine learning based systems. Recently, some security frameworks are being developed (e.g., the Adversarial ML Threat Matrix [34] by Microsoft and MITRE). The developers should obtain the latest information on the security of the machine learning techniques used in the system.

In the rest of this subsection, we refer to some research papers on the defense methods specific to machine learning.

9.3.4.1. Poisoning attacks

Defense methods for poisoning attacks (Sections 9.2.2.2.1 and 9.2.2.2.2) have been studied and proposed in terms of the detection and robustness against poisoning. For example, methods for identifying and removing outliers from the training dataset are useful to mitigate certain data poisoning attacks. Some defense methods have theoretical guarantees under strong assumptions. See Jagielski et al. [68] for defense against data poisoning attacks, and Li et al. [81] for backdoor attacks.

9.3.4.2. Evasion attacks

Defense methods for evasion attacks (Section 9.2.2.2.3) have been studied mainly from the perspective of stability (or robustness) against adversarial examples. In Section 7.6, we present methods for evaluating and improving the stability of trained models. For more information, see Goodfellow et al. [64], Krakin et al. [78], Akhtar and Mian [40].

9.3.4.3. Model extraction attacks

Defense methods for model extraction attacks (Section 9.2.2.2.4) have been discussed in the literature. Simple solutions, such as not returning confidence scores and not responding to incomplete queries, can be helpful to mitigate certain model extraction attacks. An effective generic approach is to detect model extraction attacks by observing the distributions of successive input queries to trained models. For more information, see Tramèr et al. [115] and Juuti et al. [70].

9.3.4.4. Data privacy attacks

Defense methods against data privacy attacks (Section 9.2.2.2.5) have been studied widely. Here are some example approaches to reducing the probability of successful attacks:

- Methods for mitigating the *overfitting* of trained models (e.g., regularization and dropout) can be used to mitigate the leakage of membership information to some extent.
- *Differential privacy* techniques [39] provide theoretical guarantees on membership privacy. However, since perturbations are probabilistically added, the accuracy of the trained model deteriorates. Therefore, when determining the degree of the perturbation, the developer needs to consider the tradeoffs between the membership

privacy and the model's accuracy.

– Techniques for modifying confidence scores using adversarial data have also been proposed to mitigate the leakage of membership information [69].

9.3.4.5. Attacks to cause the malfunction of interpretation/explanation functionalities

Existing attacks on the interpretation/explanation functionalities (Section 9.2.2.2.6) are effective against specific methods for interpretation/explanation, and may be defended against by using multiple methods of interpretation/explanation together.

9.4. (Informative): security check list

Table 1 lists the items that should be implemented at each development stage in line with the contents specified in the Machine Learning Quality Management Guidelines.

Table 1: Security Checklist

item number	Chapter and Section	Description of the Guidelines	Proposed security measures	Remarks
	1.3.2	lifetime-long requirement for risk assessment	Conduct a risk assessment for functional safety as part of the security risk assessment to determine high-priority items as the attack targets.	Include items such as privacy and fairness that correspond to the business in which the system will be used.
	1.3.3	when a machine-learning based system is developed by sharing works	When referencing AI and machine learning standards and guidelines, include in the contract that the entire supply chain must adhere to the referenced standards and guidelines.	Machine learning quality management guidelines, ISO/IEC standards, etc. Refer to GDPR, Cybersecurity Law of the People's Republic of China, etc., depending on the business you are designing and developing. Refer to laws related to outsourcing, such as the Act for Securing the Proper Operation of Worker Dispatching Undertakings and Improved Working Conditions for Dispatched Workers and "Act against Delay in Payment of Subcontract Proceeds, Etc. to Subcontractors" also.
	1.3.3	Security risk of contamination of learning results due to intentional inclusion of incorrect data	Gather information on attacks specific to AI and machine learning (e.g., adversarial examples and poisoning attacks in Chapter 9 of this guideline), analyze attack scenarios, and implement countermeasures against the threats and vulnerabilities. Gather information on attack and defense methods, and update defense measures every fixed periods.	The cycle of reviewing overall defense measures should be within one year. Defense measures should be prioritized and implemented from the most feasible items.

	1.5.1	Safety/risk avoidance	Identify and examine threats and vulnerabilities for risk targets. From the perspective of safety/life/economy, identify the items causing large damage, classify risks according to the magnitude of the damage, assume them as attack targets, and use them for threat and vulnerability analysis. In particular, risks that threaten fairness, privacy, and safety should be handled with higher priority.	
	1.5.2	AI Performance	Identify threats and vulnerabilities and take countermeasures assuming an attack scenario in which performance is the target of the attack.	
	1.5.3	Fairness	Use Chapter 8 to envision attack scenarios for information that could be the target of an attack (e.g., gender, skin color, etc.), identify threats and vulnerabilities, and implement countermeasures. Gather information about the business to be targeted, periodically review the items to be attacked for fairness, and update the monitoring and countermeasures.	Refer to Chapter 8 for analysis.
	1.6.2	Security/privacy	- When the nature of the information contained in the information asset is expected to change (e.g., when strong correlations between multiple pieces of information are generated or lost), monitoring and periodic countermeasures should be implemented. If sensitive relationship is generated between the information asset, the damage caused by attacks can be larger. - Make sure that the new information observed by monitoring do not violate domestic laws and regulations, e.g., Act on the Protection of Personal Information, GDPR, Cybersecurity Law of the People's Republic	(As of February 2021, NIST refers to ISO/IEC SC42.) Pay particular attention to GDPR and Cybersecurity Law of the People's Republic of China (because the regulations and sanctions are very strict.) Collect information from NIST and ISO/IEC on a regular basis. (As of February 2021, NIST refers to ISO/IEC SC42)

			of China, and the US CCPA.	
	1.6.4	Social aspects such as ethicalness	For the data to be used, consider whether the regulations and contracts regarding the acquisition and use of data for legal rights and public interest policy reasons are in compliance with laws and regulations, or whether rights and contractual adjustments are necessary.	Database Society of Japan See Kitsuregawa, Kakinuma et al.
	1.7	Internal quality characteristics	Organize the situation of internal quality characteristics in the system to be developed, and identify, organize, and counter threats and vulnerabilities by assuming the data to be attacked and the attack scenario that will cause more damage.	To check the status of the internal quality of the system to be developed, you can refer to the references in this guideline.
	1.7	Property B-1: Coverage of the data set	Assuming an attack scenario in which the integrity of the security requirements is compromised, analyze the threats and vulnerabilities by enumerating the causes that triggered the attack, and consider countermeasures.	
	1.7.1	Sufficiency of requirements analysis	It is necessary to take into account whether the inference results and learning models output by the system to be developed will contain sensitive information. When conducting BIA (Business Impact Analysis)/PIA (Privacy Impact Analysis), the system operation flow should be listed and the combination of situations should be verified (use case review).	Countermeasures are needed if there are cases in which whether the information is sensitive or not changes over time.
	1.7.1	analysis from the request side such as risk analysis/failure mode analysis and bottom-up analysis	Security risk analysis in the development stage should be conducted in parallel with other risk analysis such as functional safety.	This will be done for both traditional information security and security when focusing on AI.

	1.7.2	Combination of situations	Check threats and vulnerabilities to input and output data for each combination and consider countermeasures.	
	1.7.4	“what kind of fairness" is required	Consider countermeasures based on the assumed damage to the target system referring to Chapter 9.	We will be taking into consideration at attacks where human manipulation of data makes it impossible to ensure fairness.
	1.7.5	The data must not have been inappropriately altered (authenticity), and the data must be sufficiently new.	Among the essentials of security risk assessment, the verification of "integrity" is applied to the impact calculation of attack scenarios.	Conduct a security risk assessment of the environment in which the data used for learning is stored.
	1.7.5	data selection adequacy Adequacy of labeling	When identifying the damage caused by an attack, include attacks that violate the adequacy of data selection and the suitability of labeling.	Identify the causes of damage and apply them to the listing of threats and vulnerabilities.
	1.7.5	Reorganizing data to meet new requirements and policies	If the assumptions of the damage caused by an attack change as a result of the reorganization of data, the threat and vulnerability lists should also be revised according to the increase or decrease in damage.	Check the scope of impact of the reorganization of data.
	1.7.6	overfitting	When assuming damage, include attacks that force the attacker to overfit.	For example, adding and processing of training data by worms.
	1.8.1	Quality Inspection	Include items related to security risks in the inspection items of quality inspection and establish a process work path that returns to the upstream process for review if risk issues are found during the inspection (any inspection item).	Include non-security items in the risk category as necessary.
	1.9.1	Social Principles on Human-centric AI	Implement risk countermeasures based on regular information gathering on new attack target perspectives and damage assumptions.	Include it in the PDCA cycle for security.
	2.2.1	IEC 15408	As a framework for security risk analysis and countermeasure planning, IEC15408 is a typical example, but one that fits the requirements analysis can be selected from the IEC27000 series and IPA guidelines. Assumption of AI-specific attack scenarios, identification of threats and vulnerabilities, and countermeasure studies should be conducted.	

	2.3.1, 2.3.2	machine-learning based system structure stakeholders of development and their roles	For AI-specific attacks, identify threats and vulnerabilities specific to machine learning through attack tree, use case analysis, and information asset list and consider countermeasures.	For example, assume an attack tree for an attack on AI as illustrated in Chapter 9 and determine countermeasures and priorities according to the assumption of an attack tree for an attack targeting AI as illustrated in Chapter 9.
	Paragraph 3 of 2.3.3	Fairness	List fairness violations as security damage targets. Create an attack tree that targets the listed fairness items.	
	Paragraph 4 of 2.3.3	Attack resistance	The following are the perspectives of attack resistance in security. -To prevent attacks from occurring in the first place. -Even if the system is attacked and tampered with, the effect of the tampering will be minimized so that the actual business will not be affected. -Even if the system is attacked, the context of computation before the attack is protected, kept. When the system is attacked, all resources at the time of attack are discarded and instantly replaced with the remaining resources, so the actual business is not affected.	Consider the scale of implementation for each theme.
	Paragraph 5 of 2.3.3	Ethicalness	Confirm ethics items and examine scenarios that could cause damage (examine attack trees).	
	Paragraph 6 of 2.3.3	Robustness	Assume an attack tree that causes damage to the robustness of the system.	Robustness is translated into English as “robustness”.
	Paragraph 1 of 2.3.4	System life cycle process	Establish risk assessment (gates) at each stage of the system life cycle.	In terms of the content and scale of implementation, the items to be implemented should be done according to the order of priority with fixed period (within one year) in mind. (If it does not fit into the scale of implementation, it tends to lose substance.

	2.3.5	Terms related to the use environment	System components specialized for AI and machine learning, such as learning and inference and the environment in which learning models are stored should be addressed, as they require not only conventional information security but also requirements specific to AI/machine learning.	
	2.3.6	Terms related to data used for building machine learning	same as above	
	Paragraph 4 of 2.3.7	A form of operation to collect data and carry out additional training for machine learning during operation and to update trained machine learning models when necessary.	General information security does not assume that the data used by the system will acquire new characteristics (e.g., fairness and privacy) during operation. We will include regular monitoring of the data stored by additional learning to see if it has acquired any new characteristics and consider countermeasures if necessary.	
	3.1	Table 1: Estimation of AI safety levels for human-related risks Table 2: Estimation of AI safety levels for economic risks	Inspecting each cell in the table, process the case assumed with the large impact of the damage.	While safety may not be considered when a business solution is built using only IT systems, it is essential to consider safety in embedded systems that are safety-related such as autonomous emergency braking system.
	3.3	Fairness	Create an attack tree for cases where personal rights and assets are affected and add it to the threat and vulnerability considerations.	Particular attention should be paid to scenarios related to sequential learning.
	4.1.1 (Figure 11)	Handling of development process with several operational stages	Risk factors (e.g., hidden correlations) should be verified each time the verification of compliance with requirements is conducted for each PoC. Include a mechanism to monitor risk factors and consider countermeasures during quality inspections and operational performance monitoring.	
	4.2.1 (Figure 12)	Process model in the stage of machine learning building	Handle sensitive information during the design phase of dataset modeling and testing, check the generation of new sensitive information, analyze threats and vulnerabilities, and incorporate responses. For inference results output from iterative training and quality check/validation tasks, check the handling of sensitive information,	

			generation of new sensitive information, analyze threats and vulnerabilities, and incorporate responses.	
	4.2.1 (Figure 13)	Model of preprocessing for training	Check the handling of sensitive information and the generation of new sensitive information during each step of the pre-processing, analyze threats and vulnerabilities, and incorporate responses.	
	4.2.1.1	ML Requirements Analysis Phase	Listing up the training data as information asset and assuming the sensitive relation among the data (e.g., correlation) yielded by reorganizing and requested construction of the characteristics of the input/output data as target of damage, consider countermeasures against threats and vulnerabilities., For quality requirements that are determined based on the characteristics of the data or the specific data set itself, identify threats and vulnerabilities and consider countermeasure for them, assuming damage from an attack.	
	4.2.1.2	Training data composition phase	Listing up the dataset as an information asset and scrutinizing the damage caused by an attack with the analysis of the relation among the data attributes (e.g., correlation) and the characteristics of the data, threats and vulnerabilities should be identified and the counter measures should be considered. Scrutinizing the generation of new relationships among the data after preprocessing(e.g., fairness and privacy) and assuming the damage caused by an attack, identify threats and vulnerabilities and consider counter measure.	
	4.2.1.3	Iterative training phase	Listing up the machine learning model, hyperparameters and learning models for implementation as information assets and scrutinizing the relation among the data stored in the dataset and the specific characteristics of the data (e.g., fairness or privacy), threats and vulnerabilities should be identified and the counter measure should be considered from the assuming the damage caused by an attack. Scrutinizing the generation of new relationships among the data	

			after preprocessing(e.g., fairness and privacy) and assuming the damage caused by an attack, identify threats and vulnerabilities and consider counter measure.	
	4.2.1.4	Quality check / assurance phase	Handling test data sets, test results (including test confirmation) and data that caused erroneous inferences as information assets and assuming damage caused by falsification or theft of information threats and vulnerabilities should be identified and counter measure should be considered.	
	4.2.2	System building / integration test phase	For the parts of the system other than machine learning components, conduct a security risk assessment using the ISO/IEC 27000 series, ISO/IEC 15408 Common Criteria, or guidelines, frameworks, and tools provided by IPA and take action based on the results.	For requirements specific to business areas, consider assessment methods with reference to various ISACs or, if no ISAC has been established, the requirements of PCI DSS etc.. When life, safety, or property is involved, be careful in selecting guidelines and frameworks.
	4.3	Quality monitoring / operation phase	The relationships among data (e.g., correlations), characteristics (e.g., fairness and privacy), and hyper-parameters contained in the learning data should be considered as information assets and regularly monitored for possible damage to be addressed. The relationships and characteristics among data should also be periodically reviewed.	
	5.1.1.1	In cases where safety functions are required,	Events that are assumed to cause serious damage in the course of safety considerations will cause more damage if they become the target of an attack, so these events should be processed with the higher priority.	Examine safety related to systems using AI or machine learning.
	5.1.1.3	Examination on risk scenarios related to system use	For high-risk events, higher priority should be given to addressing them because the damage will be greater if they become the target of an attack.	Identify risks deeply related to AI and machine learning and consider the counter measure against them.

	5.1.1.4 5.1.1.5	Qualities in use	If a damage related to the characteristic measured by the quality metrics to be employed, analyze the threats and vulnerabilities and take action.	If external quality characteristics are to be used, follow the damage assumptions and threat/vulnerability analysis and counter measure against external quality characteristics.
	5.1.2	risks of physical or human damages	For high-risk events, priority should be given to addressing them because the damage will be greater if they become the target of an attack.	Identify risks deeply related to AI and machine learning and consider counter measure against them.
	5.2.1	entruster and the development entrustee should build a consensus	During consensus building, extract attack cases that are closely related to AI/machine learning and relevant to the business in question, list the security risks and select from consideration and implementation of risk countermeasures, risk transference, or risk acceptance.	Risk transference: Implementing security systems and functions of others and delegating security requirements to others (e.g., developing solutions and security measures on AWS) Risk acceptance: Accepting the security risk as a business decision and not implementing measures or transferring the risk
	5.2.2	Role clarification	Among the progress of the work process, it should be confirmed that security risk assessments are conducted at the following stages, and decisions are made on the implementation, transference and approval of measures. - Approval of domain analysis results - At the time of approval of external specifications for software - At the time of approval of operation verification specifications for software - At the time of approval of testing results for software	In particular, check the KPI items that are deeply related to AI and machine learning.
	5.2.3 3	Quality management methods during operation	The relationships among data (e.g., correlations), properties (e.g., fairness and privacy), and hyper-parameters contained in the learning data should be considered as information assets and periodically monitored for possible damage to be addressed. The relationships and properties among data should also be periodically	

			reviewed.	
	5.3	Delta development	In the case of reusing existing software components, security risk assessments need to be conducted in the context of the new system being used. This also applies to the risk measures to be implemented as a result of the security risk assessment.	
	6	Internal quality	Assuming the occurrence of the most serious damage regarding the rationale for setting and designing the quality level determined by the developer for domain analysis, case coverage, data coverage, data distribution, accuracy, robustness, software reliability and quality stability, the counter measure should be taken according to the analysis of threats and vulnerabilities with the attack scenario.	For each quality characteristic, identify threats and vulnerabilities based on the level (LV) assigned by the developer and the rationale of the level setting take counter measure.
	6.1.2.1	Attributes and their viewpoints.	If the attribute is sensitive (e.g., skin color, race), stop using this attribute and consider measures such as anonymization. By examining the correlation among the attributes and the properties (e.g., fairness or privacy) related to the attributes, identify threats and vulnerabilities according to the expected damage caused by the attack.	
	6.1.3	Safety levels	Factors that increase risk are the cause of greater damage in case of the attacker becomes the target of an attack. Therefore, threats and vulnerabilities should be identified and addressed with reference to the results of analysis according to the requirements.	
	6.2	Coverage for distinguished problem cases	Identify and respond to threats and vulnerabilities by assuming an attack that violates the designed completeness.	
	6.3	Coverage of datasets	Analyze and take measures to threats and vulnerabilities by assuming an attack scenario that compromises the integrity of the security requirements and enumerating the causes that triggered the attack.	

	6.5.2.1	Unification and scrutiny of labeling policies	Assume an attack scenario where an attack causes data to fall into a condition that violates policy of labelling and identify and take measures to the threats and vulnerabilities involved.	
	6.5.2.2	Consistency check and recheck of data sets	Assessments to be conducted in response to changing functional requirements or usage environments and the process of outsourcing work should be considered for security risk assessment.	It is recommended to include the points of view for information security and supply chain security.
	6.5.2.3	Handling the long tail and determining measurement errors and outliers	Assuming an attack scenario where the stored data falls into a state that violates the policies to handle long-tail, measurement error, and outlier, identify and take counter measures to the threats and vulnerabilities involved.	
	6.5.2.4	Addressing Data poisoning	- Address threats and vulnerabilities related to possible attack scenarios in which data is poisoned. - Consider methods for detecting data poisoning. - Address security requirements other than cyber security in the system configuration and usage scenarios of the system to be developed.	To deal with usage scenarios, it is recommended to use guidelines (provided by ISAC or ministries) according to the business model in which the developed system will be used. If no guidelines are provided, it is recommended to refer to the guidelines for finance or critical infrastructure.
	6.5.2.5	Freshness	Assuming the attack scenario considering the damage where the freshness of the training data is violated, identify and take counter measures to the threats and vulnerabilities involved.	
	6.5.2.6	Establishment of system and structure to manage the process	Establish security risk assessment rules that apply to the entire supply chain and include periodic audits in accordance with the rules and the keeping of audit trails into the contract.	
	6.5.3	Requirements for quality levels	Reflect on each of the points raised in 6.5.2 of this table.	
	6.6.2	Relative behavior of indicators	- Monitor the behavior for input included in learning dataset with the means to check for damage (e.g., tampering) affects to the learning model. Prepare for counter measure according to the estimation of the	It is recommended to keep a trail (a record of evaluation metrics) of the behavior of the training dataset with respect to the inputs included in it.

			casus of the damage (e.g., tampering) of the learning model.	
	6.7.2	Evaluate and improve stability	Monitor the behavior for input not included in dataset with the means to check for damage (e.g., tampering) affects to the learning model. Prepare for counter measure according to the estimation of the casus of the damage (e.g., tampering) of the learning model.	It is recommended to leave a trail of responses to inputs not included in the dataset for iterative training phase, quality check/assurance phase, and quality management/in-operation phase.
	6.8.1	Open source implementation	Take counter measure to the information of threats and vulnerabilities.	See CWE and CVE.
	6.9.2	quality degradation	Create a response process and system infrastructure for cases where the cause of quality degradation during operation is caused by an attack.	
	7.1.2	Estimation of risk factors	Risk items are classified according to the scale of damage and those that cause serious damage are assumed to be the target of damage by attackers and take counter measure to the threats and vulnerabilities with the listed attack scenario.	
	7.1.3	final implementation form to some extent in order to conduct such an analysis	The characteristics to be assumed should also be applied to operational monitoring, along with ways to distinguish between changes from the original business assumptions (concept drift) and attacks.	
	7.2	Coverage of distinctive problem cases	See the security measure plan in Chapter 6, "internal quality characteristics”.	
	7.3	Coverage of datasets	Refer to the security response plan in Chapter 6, "internal quality characteristics”. - Scrutinize and reflect the features or the characteristics such as hidden correlation overlooked among features adopted according to pre-flight tests in data scrutinization stage or additional tests in testing stage. - The characteristics of the features should be periodically scrutinized	

			through monitoring processes.	
	7.3	Coverage of datasets	Assuming an attack scenario in which the integrity of the security requirements is violated, analyze the threats and vulnerabilities by enumerating the causes that triggered the attack, and consider countermeasures.	
	7.4	Uniformity of the data sets	See the security measure plan in Chapter 6, "internal quality characteristics".	
	7.5.1	Data Collection Policy	See proposed security measures in Section 6.5.2.1.	
	7.5.1	Assessment of whether the requirements definition has been updated in line with the data collection policy, and whether it is necessary to reconfirm the contents confirmed in the previous steps (e.g., internal quality A-1 to B-2) in line with the update.	If policies and requirement definitions is updated, confirm the change of assumption of security damage and reflect the change to the content of the measures and the analysis of threats and vulnerabilities when needed.	
	7.6.1.3	fuzz testing	If the input data is related to threats or vulnerabilities, tests to input fuzzing data should also be conducted as necessary.	
	7.6.1.5	Automatic generation of test inputs	If necessary, include testing items that target Model Evasion, Model Extraction, and Adversarial Example which are attacks deeply related to AI and machine learning.	
	7.6.2.2	regularization	In dropout, the ratio of neurons to be deactivated is a hyperparameter. These hyperparameters which determine the network structure are listed as information assets.	Other items in chapter 7.5.2 will be examined in the same manner as described on the left.

	7.6.2.3	adversarial data generation	If necessary, assume threats and vulnerabilities from attack scenarios that assume attacks by hostile data generation and consider countermeasures.	
	7.6.2.6	adversarial training	If necessary, assume threats and vulnerabilities from attack scenarios that assume attacks using adversarial training and consider countermeasures.	
	7.7.3	Common Vulnerability Enumeration (CVE)	Although CVE and CWE are very helpful as general information security threats and vulnerabilities information, they do not include insights deeply related to AI/machine learning, so we will separately set security perspectives related to AI/machine learning according to the system to be developed and use them for testing.	
	7.7.5	Software updates	One of the advanced attack methods in recent years is related to software updates. Since attacks on machine learning components may be triggered by malware contamination and session hijacking, we will assume and verify the attack scenarios.	
	7.8.1	monitoring	Monitoring from a security perspective should also be conducted focusing on the case that the relationships among features (e.g., hidden correlation) that could cause serious damage in the event of an attack, especially in the case that the relationships are sensitive.	
	7.8.2	Concept Drift	Same as monitoring	
	7.8.3	updating trained machine learning models	Same as monitoring	
	8.1.1	Ethics and Fairness	Identify threats and vulnerabilities that lead to the cause and establish countermeasures based on the assumption of damage and attack scenarios in the event that ethics and fairness are violated.	
	8.1.2	Social demand	As for legal regulations, social principles, guidelines, and international standards, incorporate them into the assumptions of attack scenarios, assuming that damage may occur if warranties are claimed or if there is a significant impact on business. (See Chapter 8.4 for more information on attack scenario assumptions.)	

10. (informative) Information on related documents

The content of this Chapter is informative.

10.1. Relation with other guidelines

10.1.1. Contract Guidelines on AI of the Ministry of Economy, Trade and Industry

"The Contract Guidelines on Utilization of AI and Data" [28] published by the Ministry of Economy, Trade and Industry summarizes considerations concerning contracts with business partners when they cooperatively develop systems containing AI (e.g. machine learning) in accordance with contracts such as sales order or quasi-mandate.

The Contract Guidelines clarify roles and responsibilities in contracts between business partners engaged in development, while the Contract Guidelines summarize quality of service which providers have to provide to users of developed system. From the standpoint of this Guideline, qualities in use of products and services defined in this guideline are elaborated with the cooperation of all stakeholders listed in the Contract Guidelines for development, and then they are provided to users. It is out of the scope in this Guideline how to concretely realize the quality with the cooperation of business partners and how to conclude a contract and share responsibilities. Stakeholders should agree on these matters based on the Contract Guidelines. On the other hand, this Guideline can serve as a base of examining technical matters related to quality when stakeholders share responsibilities and exchange information. In Section 5.2, we briefly analyzed a possibility of sharing roles as one example.

The Contract Guidelines state that a non-waterfall model is appropriate for sales orders in the development process, while the Contract Guidelines (p.46) focus not only on (3) "Development phase" but also phases such as (4) "Additional learning phase" before and after (3). Therefore, this Guideline defines a system lifecycle process by a wide range from system planning to disposal after operation. A model mixed with waterfall model is basically used and summarized as "Figure 6: Conceptual diagram of mixed machine learning lifecycle process" in page 25. With regard to the relation with "gradually exploratory" development processes recommended in the Contract Guidelines, the developmental phase in the center of Figure 6 in this Guideline corresponds basically to the development phase in the Contract Guidelines.

10.1.2. Relations with Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence (QA4AI)

The Consortium of Quality Assurance for Artificial Intelligence-based products and service (QA4AI) in Japan has published "Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence 2020.08" [38] in August 2020. The QA4AI Guidelines propose the

following five axes to be considered in quality assurance of AI products with the aim of giving "common guidelines for quality assurance of AI products".

A) Data integrity
B) Model robustness
C) System quality
D) Process agility
E) Customer expectation

Moreover, the QA4AI Guidelines present check lists of these axes and a list of specific quality management technologies.

As regards the relation between this Guideline with the QA4AI Guidelines, the three quality-assurance axes above (A, B and C) are considered to correspond to the internal qualities listed in Section 1.7 (page 16) of this Guideline. Therefore, the technologies specifically listed in the QA4AI Guidelines correspond to the quality management method for each internal quality characteristic presented in Chapter 7 and give some insight to engineers who are responsible for realizing the quality with the complementary help of the two guidelines. Table 6 explains the specific relation between the items listed in the checklists for those three axes of the QA4AI Guidelines and the internal qualities listed in Section 1.7 of this Guideline.

Although no clear quality management axis has been established in this Guideline for the quality assurance axes (D and E above), they can be realized through the application process (Section 5.1) envisioned in this Guideline (Section 5.1) and a business process with customers included therein.

Consequently, the QA4AI Guidelines published by the Consortium are beneficial as a reference for engineers who actually create machine learning-based AI to find feasibility of technologies to improve and elaborate internal qualities of machine learning components. On the other hand, this Guideline intends to comprehensively analyze matters necessary for businesses that plan and develop overall systems containing machine learning-based AI to ensure qualities in use throughout the lifecycle process and list them as much as possible. Therefore, it is thought that these two guidelines complement each other.

Table 6: Analysis of the relation with the QA4AI Guidelines (Version of August 2020)

Internal quality characteristics in this Guideline	Checklist in the QA4AI Guidelines
A-1: Sufficiency of problem domain analysis	2.2.1 Data integrity (b) Adequacy of training data (b.i) 2.2.2 Model robustness (h) Diversity of validating data (h.i)

A-2: Coverage for distinguished problem cases	2.2.1 Data integrity (a) Sufficiency of the amount of training data (a.i) (a.ii) (a.iii) (b) Adequacy of training data (b.i)
B-1: Coverage of datasets	2.2.1 Data integrity (d) Impartiality of training data (f) Consideration of characteristics in data
B-2: Uniformity of datasets	2.2.1 Data integrity (d) Impartiality of training data
B-3: Adequacy of data	2.2.1 Data integrity (b) Adequacy of training data (b.ii) (b.iii) (c) Suitability of training data for requirements (c.ii) (e) Complexity of training data (g) Adequacy of value ranges in training data
C-1: Correctness of trained models	2.2.1 Data integrity (i) Adequacy of test data 2.2.2 Model robustness (a) Sufficiency of model accuracy (b) Sufficiency of model generalization performance (c) Sufficiency of model evaluation (d) Adequacy of learning process (e) Adequacy of model structure 2.2.3 System quality (a) Suitability of value provision by the system (a.ii)
C-2: Stability of trained models	2.2.2 Model Robustness (b) Sufficiency of model generalization performance (d) Adequacy of learning process (e) Adequacy of model structure (f) Adequacy of model testing (g) Model robustness
D-1: Reliability of underlying software systems	2.2.1 Data integrity (k) Adequacy of data processing programs 2.2.2 Model robustness (k) Adequacy of model implementation in code
E-1: Maintainability of quality in operation	2.2.1 Data integrity (j) Consideration of effects of online learning 2.2.2 Model robustness (i) Sufficiency of testing after model updates (j) Consideration of model obsolescence

	2.2.3 System quality (i) Consideration of system quality degradation
Items corresponding to external qualities	2.2.3 System quality (a) Suitability of value provision by the system (c) Adequacy of units of system evaluation (d) Mitigation of impacts due to accidents (e) Prevention of accident occurrence (f) Mitigation of AI influence

10.2. Relations with international initiatives for quality of AI

Currently, diverse international initiatives for quality of AI have been taken including those mentioned below. This Guideline adopts adaptable parts from these initiatives. Moreover, we will actively present outstanding knowledge acquired from this guideline toward international standardization.

10.2.1. Quality and safety

An examination on quality and safety has started at ISO/IEC JTC 1/SC 42/WG 3. Gaps with existing standards for quality characteristics and quality assurance technologies of AI (ISO/IEC 25000 (SQuaRE), ISO/IEC/IEEE 29119-4 (Testing technology) and functional safety (IEC 61508, ISO 26262)) have been analyzed. In addition, discussions on topics such as data quality have started at SC42.

In Europe, the European Commission published the European AI Policy Guidelines [23] in 2018, including ensuring an appropriate ethical and legal framework, and established the AI-HLEG (AI Expert Group) [25]; in 2019, the AI-HLEG published the European AI Ethics In 2020, the European Commission published the European AI White Paper [24], which requires six requirements for AI systems in high-risk industrial domains and high-risk applications, including training data and accuracy. In April 2021, following a public consultation on the European AI White Paper, the European Commission published the European AI Bill [22], which requires eight requirements for data and accuracy. The content of the European AI legislation may change in the process of discussion in the European Parliament following the Commission's initiative.

It is possible that the European AI Bill, the world's first legally binding AI hard law proposal, will ultimately establish the objectives (goals) to be protected, where this guideline can provide the technical means to achieve the goals in the future. The bill requires, in its data and data governance requirements, that high quality training, validation, and test data be used for development, and that the characteristics, properties, and elements of the geographic, behavioral, and functional use environments be taken into account. It suggests the importance of B-1: Coverage of datasets and B-2: Uniformity of datasets, corresponding in this guideline.

The bill also calls for accuracy, robustness and cybersecurity requirements to ensure consistent performance throughout the lifecycle, declared accuracy levels and standards, and cybersecurity against hostile data. It also indicates the importance of C-1: Accuracy of training models and C-2: Stability of training models in this guideline.

10.2.2. Transparency

In Europe, the European AI Ethics Guidelines and the European AI Bill have outlined requirements for transparency, which are expected to have an international impact, especially in EU member states. In April 2019, the AI-HLEG presented a checklist for ensuring transparency, which will be verified through actual demonstrations with companies, and in 2020 published a self-assessment list of trusted AI [27], including seven requirements for transparency and accountability. The European AI Bill requires transparency and provision of information to users, which means transparency of operation (design and development of transparent operation so that the user can understand and control the processing process).

On the other hand, IEEE is currently examining IEEE P7001 (transparency of autonomous systems) and it may have a certain level of influence over future standardization in terms of the definitions of terms and concept. The six transparency levels (0～5) have been defined for the five types of stakeholders (users, accident investigation committee, etc.) (a higher number does not always mean that it is stricter). The certification of its compatibility is demonstrated through the pilot validation project called ECPAIS.

ISO includes terms and concepts related to transparency in the document TR24028 [6] at WG3 (trustworthiness) of ISO/IEC JTC 1/SC 42.

10.2.3. Fairness (bias)

EU AI high-level expert group (AI-HLEG) compiles high-level principles for problems of ELSI of AI including bias. The European AI Bill calls for data and data governance requirements that data sets be relevant, representative, error-free, and have appropriate statistical properties, as well as monitoring, detection, and correction of bias.

The above IEEE P7003 (algorithmic bias) [19] specifies a method to identify *negative bias* (both legally-prohibited discrimination based on race or gender and non-legal discrimination) in the development of algorithms and keep bias within the acceptable range in the lifecycle from system planning to operation. A demonstration experiment of this method is under way in the pilot project (ECPAIS (Ethics Certification Program for Autonomous and Intelligent Systems)) to validate its conformity.

ISO/IEC JTC 1/SC 42/WG 3 (trustworthiness) is also preparing documents such as TR 24028 (Overview of trustworthiness in Artificial Intelligence), TR 24027 (Bias in AI systems and AI aided decision making).

10.2.4. Other quality aspects

Privacy [18], nudging, and other issues are being considered in the IEEE P7000 series, and governance and other issues are being considered in ISO/IEC JTC1/SC42. The European AI bill requires a risk management system, technical documentation, record-keeping, and human oversight, in addition to the requirements already mentioned. In addition to the existing requirements, the report also calls for a risk management system, technical documentation, record-keeping, and human oversight.

11. (informative) Analytical information

This Chapter sorts out a process of analysis to draw out the characteristic axes of internal qualities listed mainly in Section 1.7 and Chapter 6 as reference information.

The content of this Chapter is informative.

11.1. Analysis of characteristic axes of internal qualities with respect to risk avoidance

First, a quality deterioration mode was identified with regard to external quality axes of risk avoidance. This quality deterioration occurs when *individual inference results of machine learning components (to describe it in an extreme manner, vector values of neural-network results) is not correct, favorable or desirable*. Thus, we analyzed a possibility that a machine learning component gives *undesirable* answers based on a concept similar to Fault Tree Analysis (FTA) using an abstract and binary tree failure mode on the assumption of abstract machine learning components. The analysis results are shown in Figure 21.

This analysis aims to comprehensively decompose the causes of failure modes. It should be noted that, when any misjudgment was made, it might be impossible to identify the cause without so-called oracle perspective. On the other hand, it is possible to reduce a possibility of all failure modes by taking measures for all causes of failure without oracle perspective. Of course, it is impossible to perfectly realize each paragraph. Moreover, errors included in training input data and various gaps such as mathematical issues in super-multidimensional space data are ignored intentionally in this analysis. However, if these items are focused on as a direction of the overall quality improvement process, it can make a sufficient contribution to the improvement of quality despite gaps.

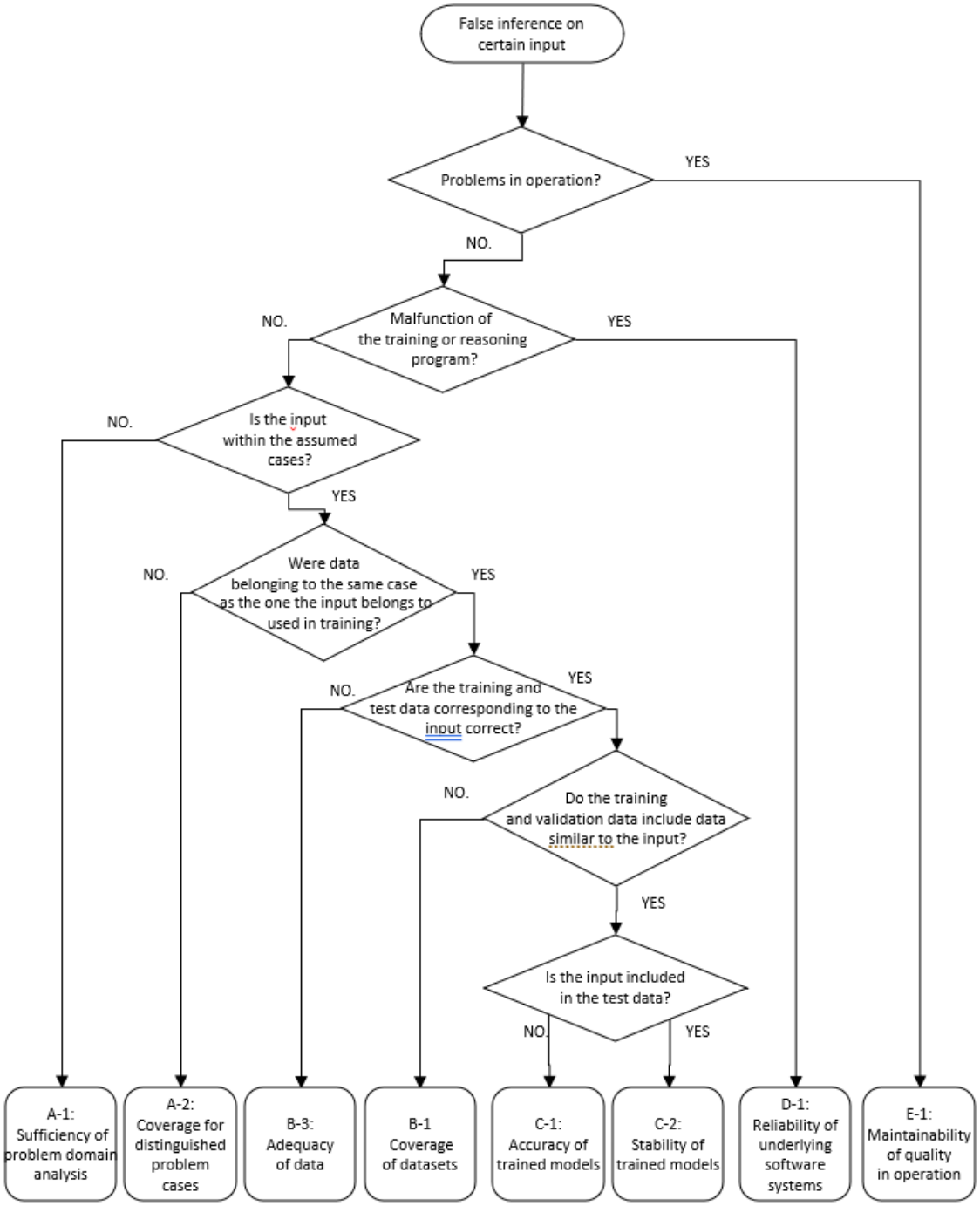

Figure 21: Example of analysis of failure mode with respect to machine learning

11.2. Analysis of quality management axes with respect to AI performance

AI performance was analyzed in a similar way based on the internal quality management axes with respect to *risk avoidance* mentioned in the previous section.

Strictly speaking, AI performance also deteriorates, when *individual inference results of*

machine learning components (to describe in an extreme manner, vector values of neural-network results) is not correct, favorable or desirable. A difference from risk avoidance is a difference in overall evaluation functions caused by different weighting of individual cases. Therefore, the same internal quality characteristic axes can be basically used as they are.

However, risk avoidance focuses on sufficient assignment of training data as measures for each anticipated risk case and does not take into account the balance of overall training data intentionally. This is because sufficient learning cannot be achieved by uniformly sampling training data especially from serious risk cases, although their frequency of occurrence is very low. From the viewpoint of AI performance, however, it is widely known that this type of *intentionally-biased reinforced learning data* may cause deterioration of overall performance. That is why we added *Uniformity of training datasets* listed in 6.4 as an internal quality axis which mainly envisions AI performance. The results are nine internal characteristics listed in Chapter 6.

12. Tables and figures

The content of this Chapter is informative.

12.1. Tables of relations between external quality characteristics and internal quality characteristics

The numbers and symbols in the tables refer to the levels of check items listed in *Requirements for quality levels* of applicable sections. The symbol "+" means that additional examinations will be made in the future. The bolded items mean that they require special attention.

AISL	0	0.1	0.2	1	2	3	4
A-1 Sufficiency of problem domain analysis		1	2	3	+	+	+
A-2 Coverage for distinguished problem cases		1	2	3	+	+	+
B-1 Coverage of datasets		1	2	3	+	+	+
B-2 Uniformity of datasets		S1	S2	S2	+	+	+
B-3 Adequacy of data		1	2	3	+	+	+
C-1 Correctness of trained models		1	2	3	+	+	+
C-2 Stability of trained models		1	2	3	+	+	+
D-1 Reliability of underlying software system		1	2	3	+	+	+
E-1 Maintainability of quality in operation		1	2	3	+	+	+

AIPL/AIFL	PL 0	PL 1	PL 2	FL 0	FL 1	FL 2
A-1 Sufficiency of problem domain analysis		1	2		1	2
A-2 Coverage for distinguished problem cases		1	2		1	2
B-1 Coverage of datasets		1	2		1	2
B-2 Uniformity of datasets		E1	E2		***E2***	E2
B-3 Adequacy of data		1	2		1	2
C-1 Correctness of trained models		1	2		1	2
C-2 Stability of trained models		1	2		1	2
D-1 Reliability of underlying software system		1	2		1	2
E-1 Maintainability of quality in operation		1	2		2	3

13. Bibliography

Japanese titles of literatures and organization names which have official English translations are shown in English. Those which does not have official translations are shown both in Japanese and with unofficial translations.

13.1. International standards

[1] ISO 13849-1:2015: Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design.

[2] ISO/IEC/IEEE 15288:2015: Systems and software engineering — System life cycle processes.

[3] ISO/IEC 15408-1:2009: Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model.

[4] ISO/IEC DIS 22989: Information technology — Artificial intelligence — Artificial intelligence concepts and terminology.

[5] ISO/IEC DTR 24027: Information technology — Artificial Intelligence (AI) — Bias in AI systems and AI aided decision making.

[6] ISO/IEC TR 24028:2020: Information technology — Artificial intelligence — Overview of trustworthiness in artificial intelligence.

[7] ISO/IEC 25010:2011: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

[8] ISO/IEC 25012:2008: Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model.

[9] ISO 26262-1:2018: Road vehicles — Functional safety — Part 1: Vocabulary.

[10] ISO/IEC 27000:2018: Information technology — Security techniques — Information security management systems — Overview and vocabulary.

[11] ISO/IEC/IEEE 29119-4:2015: Software and systems engineering — Software testing — Part 4: Test techniques.

[12] IEC 61508-1:2010: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements.

[13] IEC 61508-3:2010: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements.

[14] IEC 61508-4:2010: Functional safety of electrical/electronic/programmable

electronic safety-related systems - Part 4: Definitions and abbreviations.

[15] IEC 62278:2002: Railway applications - Specification and demonstration of reliability, availability, maintainability and safety (RAMS).

[16] IEC TS 62998-1:2019: Safety of machinery - Safety-related sensors used for the protection of persons.

[17] IEEE P7001 - IEEE Draft Standard for Transparency of Autonomous Systems.

[18] IEEE P7002 - IEEE Draft Standard for Data Privacy Process.

[19] IEEE P7003 - Algorithmic Bias Considerations. An active project.

13.2. Documents/regulations from countries and international organizations

[20] Council for Science, Technology and Innovation, Cabinet Office of Japan. Social Principles of Human-Centric AI. March 2019. https://www8.cao.go.jp/cstp/aigensoku.pdf (In Japanese), https://www.cas.go.jp/jp/seisaku/jinkouchinou/pdf/humancentricai.pdf (tentative translation).

[21] Organisation for Economic Co-operation and Development (OECD). *OECD Principles on Artificial Intelligence*. May 2019. https://www.oecd.org/going-digital/ai/principles/

[22] European Commission. *Regulation of the European parliament and of the council laying down harmonized rules on artificial intelligence (Artificial Intelligence Act) and amending certain union legislative acts*. April 2021. https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence-artificial-intelligence

[23] European Commission. *Communication Artificial Intelligence for Europe*. 2018. https://digital-strategy.ec.europa.eu/en/library/communication-artificial-intelligence-europe.

[24] European Commission. *The White Paper on Artificial Intelligence – A European approach to excellence and trust*. February 2020. https://ec.europa.eu/info/publications/white-paper-artificial-intelligence-european-approach-excellence-and-trust_en

[25] European Commission. High-Level Expert Group on Artificial Intelligence. 2018. https://digital-strategy.ec.europa.eu/en/policies/expert-group-ai

[26] The High-Level Expert Group on Artificial Intelligence, European Commission. *Ethics guidelines for trustworthy AI*. April 2019. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai

[27] The High-Level Expert Group on Artificial Intelligence, European Commission. *The Assessment List for Trustworthy Artificial Intelligence (ALTAI)*. July 2020. https://ec.europa.eu/digital-single-market/en/news/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment

[28] Ministry of Economy, Trade and Industry, Japan. Contract Guidelines on Utilization of AI and Data. June 2018. In Japanese, https://www.meti.go.jp/press/2018/06/20180615001/20180615001-3.pdf

[29] AI 社会実装アーキテクチャ検討会 [Study group for architecture of AI social implementation], Ministry of Economy, Trade and Industry, Japan. 我が国の AI ガバナンスの在り方 ver1.0 [AI Governance of our country - version 1.0]. January 2021. In Japanese, https://www.meti.go.jp/press/2020/01/20210115003/20210115003-1.pdf

[30] The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems, First Edition*. IEEE, 2019. https://standards.ieee.org/content/ieee-standards/en/industry-connections/ec/autonomous-systems.html

13.3. Official standards and forum standards

[31] National Institute of Standards and Technology (United States of America). Draft NIST IR 8269: *A Taxonomy and Terminology of Adversarial Machine Learning*. October 2019. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf

[32] National Institute of Standards and Technology (United States of America). NIST SP 800-30: *Guide for Conducting Risk Assessments*. September 2012. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf

[33] National Institute of Standards and Technology (United States of America). *Cyber Security Framework Version 1.1*. April 2018. https://doi.org/10.6028/NIST.CSWP.04162018

[34] MITRE. *Adversarial ML Threat Matrix*. https://github.com/mitre/advmlthreatmatrix

[35] MITRE, *Common Platform Enumerations*. http://cpe.mitre.org/

[36] MITRE, *Common Vulnerability Enumerations*. https://cve.mitre.org/

[37] Payment Card Industry Security Standards Council. Payment Card Industry Data Security Standard (PCI DSS). https://www.pcisecuritystandards.org/.

[38] Consortium of Quality Assurance for Artificial-Intelligence-based products and services (QA4AI). Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence – 2020.08 version. August 2020. http://qa4ai.jp/download/.

13.4. Research Articles

[39] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proc. of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pp. 308–318, 2016.

[40] Naveed Akhtar, and Ajmal Mian. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* 6, pp. 14410–14430, 2018. https://arxiv.org/pdf/1801.00553.pdf

[41] P. Ammann, and J. Offutt. *Introduction to Software Testing*, Cambridge University Press, 2008.

[42] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[43] Guy Barash, Eitan Farchi, Ilan Jayaraman, Orna Raz, Rachel Tzoref-Brill, and Marcel Zalmanovici. Bridging the gap between ML solutions and their business requirements using feature interactions. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019)*, pp. 1048–1058, August 2019. https://dl.acm.org/citation.cfm?id=3340442

[44] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The Oracle Problem in Software Testing: A Survey. In *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015.

[45] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pp.3240–3247, 2019.

[46] T. Byun, V. Sharma, A. Vijayakumar, S. Rayadurgam, and D. Cofer, Input Prioritization for Testing Neural Networks, Proc. AITest, pp. 63-70, and arXiv:1901.03768

[47] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Optimized Pre-Processing for Discrimination Prevention. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, December 2017. https://papers.nips.cc/paper/6988-optimized-pre-processing-for-discrimination-prevention.pdf.

[48] Nicholas Carlini, and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of the 38th IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

[49] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy and Biplav Srivastava. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *Proceedings of the Workshop on Artificial Intelligence Safety 2019*, 2019. http://ceur-ws.org/Vol-2301/paper_18.pdf

[50] T. Y. Chen, S. C. Chung, and S. M. You. Metamorphic Testing – A New Approach for Generating Next Test Cases, Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology, 1998.

[51] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, Y. H. Tse, and Z. Q. Zhou. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys*, 51(1):1–27, 2018.

[52] S. Chiappa, and W. S. Isaac. A Causal Bayesian Networks: Viewpoint on Fairness. In *Privacy and Identity Management: Fairness, Accountability, and Transparency in the Age of Big Data*, IFIP Advances in Information and Communication Technology, vol. 547, 2019.

[53] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. *The 36th International Conference on Machine Learning (ICML 2019)*, 2019.

[54] George Deckert, NASA Hazard Analysis Process. Johnson Space Center, National Aeronautics and Space Administration, 2010. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100040678.pdf.

[55] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher J. Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2019 (NeurIPS'19)*, pp. 13567–13578, 2019. https://arxiv.org/pdf/1906.07983.pdf

[56] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jin Song Dong, and Dai Ting. There is Limited Correlation between Coverage and Robustness for Deep Neural Networks. arXiv:1911.05904 [cs.LG]. https://arxiv.org/abs/1911.05904

[57] S. Elbaum and D. S. Rosenblum. Known Unknowns – Testing in the Presence of Uncertainty, In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*, pp. 833–836, 2014.

[58] Arisa Ema (ed.). Small Featured Articles "From AI principles to implementation: introduction of international artivities" [AI 原則から実践へ：国際的な活動紹介]. Artificial Intelligence, 36(2), The Japan Society for Artificial Intelligence, March 2021. In Japanese.

[59] E. R. Faria, J. Gama, and A. C. Carvalho. Novelty detection algorithm for data streams multi class problems, In *Procedings of the 28th annual ACM symposium on*

applied computing, pp. 795–800, 2013.

[60] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and Z. Chen, DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks, Proc. 29 th ISSTA, pp. 177-188, and arXiv:1903.00661v2, 2020.

[61] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15),* pp. 1322–1333, 2015. https://dl.acm.org/doi/10.1145/2810103.2813677

[62] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, A survey on concept drift adaptation. In *ACM computing surveys*, 46(4):1–37, April 2014.

[63] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pp. 2672–2680, 2014.

[64] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proc. of the 3rd International Conference on Learning Representations (ICLR'15)*, 2015. https://arxiv.org/pdf/1412.6572.pdf

[65] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim. Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks? In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*, pp. 851–862, 2020.

[66] Christina Ilvento. Metric Learning for Individual Fairness. arXiv:1906.00250 [cs.LG]. https://arxiv.org/abs/1906.00250

[67] L. Inozemtseva, and R. Holmes. Coverage is not Strongly Correlated with Test Suite Effectiveness, In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, pp. 435–455, 2014.

[68] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru and Bo Li. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P'18)*, pp. 19–35, 2018. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8418594.

[69] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adver- sarial Examples. In *Proc. of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, pp. 259–274, 2019.

[70] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: Protecting Against DNN Model Stealing Attacks. In *Proc. of the IEEE European Symposium on Security and Privacy (Euro S&P'19)*, pp.512–527, 2019. https://arxiv.org/pdf/1805.02628.pdf

[71] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *International Conference on Computer-Aided Verification (CAV)*, 2017.

[72] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, 7524:35–50, 2012. https://link.springer.com/content/pdf/10.1007%2F978-3-642-33486-3_3.pdf

[73] J. Kim, R. Feldt, and S. Yoo. Guiding Deep Learning System Testing Using Surprise Adequacy, In *Proceedings of the 41st International Conference on Software Engineering (ICSE '19)*, pp. 1039–1049, 2019.

[74] Rob Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *The 14th International Joint Conference on Artificial Intelligence*, 2(12):1137–1143, 1995.

[75] F. Kamiran, A. Karim, and X. Zhang. Decision theory for discrimination-aware classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2012)*, 2012.

[76] Faisal Kamiran, and Toon Calders. Data preprocessing techniques for classification without discrimination. In *Knowledge and Information Systems*, 33(1):1–33, 2012.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Procedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012*), pp.1097–1105, 2012.

[78] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. In *Proc. of the 5th International Conference on Learning Representations (ICLR) Workshop*, arXiv:1607.02533 [cs.CV], July 2016, https://arxiv.org/pdf/1607.02533.pdf

[79] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified Robustness to Adversarial Examples with Differential Privacy. *The IEEE Symposium on Security and Privacy* (SP), 2019.

[80] S. Lee, J. Kim, J. Jun, J. Ha, and B. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Proc. of the Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 4652–4662, 2017.

[81] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor Learning: A Survey. arXiv:2007.08745 [cs.CR], July 2020, https://arxiv.org/pdf/2007.08745

[82] P. Lindstrom, B. M. Namee, and S. J. Delany. Drift detection using uncertainty distribution divergence. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 604–608, 2011.

[83] X. Liu, M. Cheng, H. Zhang, and C. Hsieh. Towards robust neural networks via

random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV 2018)*, 2018.

[84] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems, In *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2018)*, pp. 120–131, 2018.

[85] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *Network and Distributed Systems Security Symposium (NDSS)*, 2019.

[86] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *the Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

[87] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A Survey on Bias and Fairness in Machine Learning. arXiv:1908.09635 [cs.LG], 2019.

[88] B.P. Miller, L. Fredricksen, and B. So. An Empirical Study of the Reliability of UNIX Utilities, *Communications of the ACM*, 33(12):32–44, 1990.

[89] Shin Nakajima, Software Testing under Dataset Diversity, Computer Software, 35(2):26-32, 2018 [中島 震. データセット多様性のソフトウェア・テスティング, コンピュータ・ソフトウェア, 35(2):26–32, 2018]. In Japanese.

[90] S. Nakajima: Dataset Diversity for Metamorphic Testing of Machine Learning Software, In Proc. 8th SOFL+MSVL, pp. 21-38, 2019.

[91] Shin Nakajima. Distortion and Faults in Machine Learning Software, In *Proc. 9th International Workshop on SOFL + MSVL for Reliability and Security*, pp. 29–41, 2020, and also arXiv:1911.11596 [cs.LG], 2019.

[92] Shin Nakajima, "Quality Issues in Machine Learning from Software Engineering Viewpoints", Maruzen Publisher, ISBN 978-4-612-30573-7, November 2020. [中島 震. ソフトウェア工学から学ぶ機械学習の品質問題, 丸善出版, 2020]. In Japanese.

[93] Shin Nakajima, Distortion Metrics for Trained Machine Learning Models, IEICE Technical Report, SS2019-44, March 2020. [中島 震. 訓練済み機械学習モデル歪みの定量指標, 電子情報通信学会ソフトウェアサイエンス研究会, 2020]. In Japanese.

[94] Shin Nakajima. Statistical Partial Oracle for Machine Learning Software Testing, In *Proceedings of the 10th International Workshop on SOFL + MSVL for Reliability and Security*, 2021.

[95] Shin Nakajima and Tsong Yueh Chen. Generating Biased Dataset for Metamorphic Testing of Machine Learning Programs, In *Proceedings of The 31st IFIP International Conference On Testing Software And Systems (IFIP–ICTSS 2019)*, pp. 56–64, 2019.

[96] Takako Nakatani, Shin Nakajima “Software Engineering”, Foundation for the Promotion of the Open University of Japan, ISBN 978-4-595-14119-5, March 2019 [中谷 多哉子, 中島 震. ソフトウェア工学. 放送大学大学院教材, 放送大学教育振興会, 2019 年 3 月]. In Japanese.

[97] Steven Nowlan, and Geoffrey Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4), 1992.

[98] L. Oneto, N. Navarin, A. Sperduti, and D. Anguita. Recent Trends in Learning From Data. *Tutorials from the INNS Big Data and Deep Learning Conference (INNSBDDL2019)*, January 2020.

[99] Tinghui Ouyang, Vicent Sanz Marco, Yoshinao Isobe, Hideki Asoh, Yutaka Oiwa, and Yoshiki Seo. Corner Case Data Description and Detection. arXiv:2101.02494v2 [cs.LG]. https://arxiv.org/pdf/2101.02494.pdf

[100] Judea Pearl. Understanding Simpson’s Paradox. *The American Statistician*, 68(1):8–13, February 2014. Also UCLA Cognitive Systems Laboratory Technical Report R-414, University of California, Los Angeles, December 2013.

[101] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems, *The 26th Symposium on Operating Systems Principles (SOSP 2017)*, pp. 1–18, 2017.

[102] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger. On fairness and calibration. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS ’17)*, December 2017.

[103] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista. Fast unsupervised online drift detection using incremental Kolmogorov Smirnov test. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1545–1554, 2016.

[104] G. Rothermel, R.H. Untch, and M.J. Harrold, Prioritizing Test Cases for Regression Testing, IEEE TSE 27(10), pp. 929-948, 2001.

[105] T. S. Sethi, and M. Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

[106] B. Settles. Active Learning Literature Survey. Technical Report #1648, Computer Science Department, University of Wisconsin-Madison, 2009.

[107] S. Shalev-Shwartz. Online learning and online convex optimization, *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[108] Reza Shokri, Marco Stronati, Congzheng Song and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P’17)*, pp. 3–18, 2017. https://arxiv.org/pdf/1610.05820.pdf

[109] M. S. R. Shuvo and D. Alhadidi. Membership Inference Attacks: Analysis and Mitigation. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications* (TrustCom), pp. 1410–1419, 2020. doi: 10.1109/TrustCom50675.2020.00190. https://ieeexplore.ieee.org/document/9343081

[110] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *Proc. of the AAAI/ACM Conference on AI, Ethics, and Society 2020 (AIES'20)*, pp.180–186, 2020. https://arxiv.org/pdf/1911.02508.pdf

[111] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research* 15(56):1929–1958, 2014.

[112] H. Tian, M. Yu, and W. Wang. Continuum: A platform for cost aware, low latency continual learning, In *Proceedings of the ACM Symposium on Cloud Computing*. pp. 26–40, 2018.

[113] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering (ICSE 2018)*, pp. 303–314, 2018.

[114] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

[115] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security '16)*. https://arxiv.org/pdf/1609.02943.pdf

[116] Guillermo Valle-Pérez and Ard A. Louis. Generalization bounds for deep learning. arXiv:2012.04115v2, 2020. https://arxiv.org/abs/2012.04115

[117] A. Vostrikov and S. Chernyshev. Training sample generation software. In *Intelligent Decision Technologies 2019*, Smart Innovation, Systems and Technologies (SIST), 143:145–151, 2019.

[118] J. Wang, J. Chen, Y. Sun, X. Ma, D. Wang, J. Sun, and P. Cheng. RoBOT: Robustness-Oriented Testing for Deep Learning Systems, In *Proceedings of the 43rd International Conference on Software Engineering (ICSE 2021)*, 2021.

[119] Tsui-Wei Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, PMLR 97:6727–6736, 2019.

[120] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S. Dhillon, and Luca Daniel. Towards Fast Computation of Certified

Robustness for ReLU Networks. In *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80:5276-5285, 2018.

[121] E. J. Weyuker. On Testing Non-testable Programs, Computer Journal, 25(4):465–470, 1982.

[122] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *The 35th International Conference on Machine Learning (ICML 2018)*, PMLR vol. 80, pp. 5283–5292, 2018.

[123] Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees. In *Theoretical Computer Science,* 2019. https://doi.org/10.1016/j.tcs.2019.05.046

[124] Weilin Xu, David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Network and Distributed Systems Security Symposium* (NDSS), 2018.

[125] B. H. Zhang, B. Lemoine, and M. Mitchell. Mitigating Unwanted Biases with Adversarial Learning. In *AIES '18: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, December 2018. https://dl.acm.org/doi/pdf/10.1145/3278721.3278779.

[126] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine Learning Testing: Survey, Landscapes and Horizons. arXiv:1906.10742 [cs.LG]. https://arxiv.org/abs/1906.10742

[127] P. Zhang, Q. Dai, and P. Pelliccione. CAGFuzz: Coverage-Guided Adversarial Generative Fuzzing Testing of Deep Learning Systems. arXiv:1911.07931, 2019.

[128] P. Zhang, J. Wang, J. Sun, G. Dong, and X. Wang. White-box Fairness Testing through Adversarial Sampling. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020)*, pp. 1–12, 2020.

13.5. Miscellaneous

[129] Information-processing Promotion Agency, Japan (IPA). はじめての *STAMP/STPA* [STAMP/STPA for beginners]. June 2019. In Japanese, https://www.ipa.go.jp/ikc/reports/20190329.html

[130] Security Center, Information-processing Promotion Agency, Japan (IPA). つながる世界の品質確保に向けた手引き [Guidance for quality assurance for connected worlds]. June 2018. In Japanese, https://www.ipa.go.jp/sec/publish/tn18-001.html

[131] Security Center, Information-processing Promotion Agency, Japan (IPA). 共通プラットフォーム一覧 CPE 概説 [Overview of Common Platform Enumeration (CPE)]. October 2008, In Japanese, https://www.ipa.go.jp/security/vuln/CPE.html

[132] Security Center, Information-processing Promotion Agency, Japan (IPA). 共通脆弱性識別子 CVE 概説 [Overview of Common Vulnerability Enumeration (CVE)]. January 2009. In Japanese, https://www.ipa.go.jp/security/vuln/CVE.html

[133] National Institute of Advanced Industrial Science and Technology. Report on quality evaluation/improvement technology. Technical Report, Digital Architecture Research Center / Cyber Physical Security Research Center / Artificial Intelligence Research Center, Digiarc-TR-2022-02 / CPSEC-TR-2022003, 2022.

[134] IBM Corporation. AI Fairness 360 - Resources. http://aif360.mybluemix.net/resources

14. Changes

14.1. The second edition (July 2021)

- The definition of fairness in Section 1.5.3 has been updated based on further detailed considerations.
- Internal qualities shown in Section 1.7 and Chapter 6 have been categorized into four groups from A to D and given identifiers ranging from A-1 to E-1.
- A part of C-1 *Correctness of trained models* has been extracted, extended, and made into a new internal quality, B-3 *Adequacy of data*.
- Some internal qualities have been given new names.
- Chapter 8 “Fairness” and Chapter 9 “Security” have been added.

14.2. The English version of the second edition (Feb. 2022)

- Section 7.6.1 has been reorganized and extended with a new section, Section 7.6.1.7.
- Four new items have been added to the bibliography and cited in Section 7.6.1.

Editors and Authors

Digital Architecture Research Center (DigiARC)/
Artificial Intelligence Research Center (AIRC)/
Cyber Physical Security Research Center (CPSEC),
National Institute of Advanced Industrial Science and Technology (AIST)

List of members of the Committee for machine learning quality management

Shin Nakajima	National Institute of Advanced Industrial Science and Technology/ National Institute of Informatics (chair)
Yoshiko Seo	National Institute of Advanced Industrial Science and Technology (vice-chair)
Takashi Egawa	NEC Corporation
Shintaro Fukushima	Toyota Motor Corporation
Chinami Hamatani	Ad-Sol Nissin Corporation
Kenichi Kobayashi	Fujitsu Laboratories Limited
Hiroshi Kuwajima	Denso Corporation
Yusei Nakashima	TechMatrix Corporation
Hideto Ogawa	Hitachi, Ltd.
Yutaka Oiwa	National Institute of Advanced Industrial Science and Technology
Tamao Okamoto	Panasonic Corporation
Naoto Sato	Hitachi, Ltd.
Tomomichi Suzuki	Tokyo University of Science
Shingo Takada	Keio University
Satoshi Tsuchiya	Fujitsu Limited
Naoya Wakamatsu	NEC Corporation
Atsushi Yamada	IBM Japan, Ltd.

List of authors of the Machine Learning Quality Management Guideline

- Japanese edition:
  - Overall structure, writing and editing: Yutaka Oiwa (AIST)
  - Section 5.2: Chinami Hamatani (Ad-Sol Nisshin Corporation)
  - Section 7.6: Shin Nakajima (National Institute of Informatics)
  - Section 7.6.2: Yoshinao Isobe (AIST)
  - Section 7.8: Kenichi Kobayashi and Yoshihiro Okawa (Fujitsu Laboratories Limited)
  - Section 8: Yutaka Oiwa, (AIST), Chinami Hamatani (Ad-Sol Nisshin Corporation)
  - Sections 9.2, 9,3: Yutaka Oiwa, Yusuke Kawamoto (AIST), Kazumasa Miyake (SEI)
  - Section 9.4: Kazumasa Miyake (SEI)
  - Section 10.2: Contribution from Egawa Takashi (NEC Corporation) and Hiroshi Kuwajima (Denso)
  - Supervision: Members of
    - Committee for machine learning quality management
    - Guideline Taskforce
- English edition: members of the Guideline Taskforce
  editors: Yutaka Oiwa and Koichi Konishi (AIST)

List of members of the Guideline Taskforce (FY 2020/2021)

Yutaka Oiwa	National Institute of Advanced Industrial Science and Technology (leader and co-editor)
Koichi Konishi	National Institute of Advanced Industrial Science and Technology
Takashi Egawa	NEC Corporation
Shin Nakajima	National Institute of Informatics
Shintaro Fukushima	Toyota Motor Corporation
Chinami Hamatani	Ad-Sol Nissin Corporation
Yoshinao Isobe	National Institute of Advanced Industrial Science and Technology
Yusuke Kawamoto	National Institute of Advanced Industrial Science and Technology
Kenichi Kobayashi	Fujitsu Laboratories Ltd.
Hiroshi Kuwajima	Denso Corporation
Yosuke Motohashi	NEC Corporation
Kazumasa Miyake	Sumitomo Electric Industries, Ltd.
Takeshi Miyake	Cyber Creative Institute Company Limited
Yusei Nakashima	Techmatrix Corporation
Tamao Okamoto	Panasonic Corporation
Yoshiki Seo	National Institute of Advanced Industrial Science and Technology
Satoshi Tsuchiya	Fujitsu Limited