भारतीय मानक **IS/ISO/IEC TS 4213 : 2022**

Indian Standard

सूचना प्रौद्योगिकी — आर्टिफिशियल इंटेलीजेंस — मशीन लर्निंग वर्गीकरण प्रदर्शन का आकलन

Information Technology — Artificial Intelligence — Assessment of Machine Learning Classification Performance

ICS 35.020

भारतीय मानक ब्यूरो
BUREAU OF INDIAN STANDARDS
मानक भवन, 9 बहादुर शाह ज़फर मार्ग, नई दिल्ली - 110002
MANAK BHAVAN, 9 BAHADUR SHAH ZAFAR MARG
NEW DELHI - 110002
www.bis.gov.in www.standardsbis.in

October 2023 **Price Group 12**

Artificial Intelligence Sectional Committee, LITD 30

NATIONAL FOREWORD

This Indian Standard which is identical to ISO/IEC TS 4213 : 2022 'Information technology — Artificial intelligence — Assessment of machine learning classification performance' issued by the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) was adopted by the Bureau of Indian Standards on the recommendations of the Artificial Intelligence Sectional Committee and approval of the Electronics and Information Technology Division Council.

The text of ISO/IEC standard has been approved as suitable for publication as an Indian Standard without deviations. Certain conventions are however not identical to those used in Indian Standards. Attention is particularly drawn to the following:

a) Wherever the words 'International Standard' appears referring to this standard, they should be read as 'Indian Standard'; and
b) Comma (,) has been used as a decimal marker while in Indian Standards, the current practice is to use a point (.) as the decimal marker.

The Committee has reviewed the provisions of following International Standards referred in this adopted standard and has decided that they are acceptable for use in conjunction with this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document applies, including any corrigenda and amendment:

International Standards	*Title*
ISO/IEC 22989 : 2022	Information technology — Artificial intelligence — Artificial intelligence concepts and terminology
ISO/IEC 23053 : 2022	Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)

Contents

Page

Introduction

As academic, commercial and governmental researchers continue to improve machine learning models, consistent approaches and methods should be applied to machine learning classification performance assessment.

Advances in machine learning are often reported in terms of improved performance relative to the state of the art or a reasonable baseline. The choice of an appropriate metric to assess machine learning model classification performance depends on the use case and domain constraints. Further, the chosen metric can differ from the metric used during training. Machine learning model classification performance can be represented through the following examples:

— A new model achieves 97,8 % classification accuracy on a dataset where the state-of-the-art model achieves just 96,2 % accuracy.

— A new model achieves classification accuracy equivalent to the state of the art but requires much less training data than state-of-the-art approaches.

— A new model generates inferences 100x faster than state-of-the-art models while maintaining equivalent accuracy.

To determine whether these assertions are meaningful, aspects of machine learning classification performance assessment including model implementation, dataset composition and results calculation are taken into consideration. This document describes approaches and methods to ensure the relevance, legitimacy and extensibility of machine learning classification performance assertions.

Various AI stakeholder roles as defined in ISO/IEC 22989:2022, 5.17 can take advantage of the approaches and methods described in this document. For example, AI developers can use the approaches and methods when evaluating ML models.

Methodological controls are put in place when assessing machine learning performance to ensure that results are fair and representative. Examples of these controls include establishing computational environments, selecting and preparing datasets, and limiting leakage that potentially leads to misleading classification results. Clause 5 addresses this topic.

Merely reporting performance in terms of accuracy can be inappropriate depending on the characteristics of training data and input data. If a classifier is susceptible to majority class classification, grossly unbalanced training data can overstate accuracy by representing the prior probabilities of the majority class. Additional measurements that reflect more subtle aspects of machine learning classification performance, such as macro-averaged metrics, are at times more appropriate. Further, different types of machine learning classification, such as binary, multi-class and multi-label, are associated with specific performance metrics. In addition to these metrics, aspects of classification performance such as computational complexity, latency, throughput and efficiency can be relevant. Clause 6 addresses these topics.

Complications can arise as a result of the distribution of training data. Statistical tests of significance are undertaken to establish the conditions under which machine learning classification performance differs meaningfully. Specific training, validation and test methodologies are used in machine learning model development to address the range of potential scenarios. Clause 7 addresses these topics.

Annex A illustrates calculation of multi-class classification performance, using examples of positive and negative classifications. Annex B illustrates a receiver operating characteristic (ROC) curve derived from example data in Annex A.

Annex C summarizes results from machine learning classification benchmark tests.

Annex D discusses a chance-corrected cause-specific mortality fraction, a machine learning classification use case. Apart from these, this document does not address any issues related to benchmarking, applications or use cases.

Indian Standard

INFORMATION TECHNOLOGY — ARTIFICIAL INTELLIGENCE — ASSESSMENT OF MACHINE LEARNING CLASSIFICATION PERFORMACE

1 Scope

This document specifies methodologies for measuring classification performance of machine learning models, systems and algorithms.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 22989:2022, *Information technology — Artificial intelligence — Artificial intelligence concepts and terminology*

ISO/IEC 23053:2022, *Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)*

3 Terms and definitions

For the purposes of this document, the terms and definitions in ISO/IEC 22989:2022, ISO/IEC 23053:2022, and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1 Classification and related terms

3.1.1
classification
method of structuring a defined type of item (objects or documents) into classes and subclasses in accordance with their characteristics

[SOURCE: ISO 7200:2004, 3.1]

3.1.2
classifier
trained model and its associated mechanism used to perform *classification* (3.1.1)

3.2 Metrics and related terms

3.2.1
evaluation
process of comparing the *classification* (3.1.1) predictions made by the model on data to the actual labels in the data

3.2.2
false negative
miss
type II error
F_N
sample wrongly classified as negative

3.2.3
false positive
false alarm
type I error
F_P
sample wrongly classified as positive

3.2.4
true positive
T_P
sample correctly classified as positive

3.2.5
true negative
T_N
sample correctly classified as negative

3.2.6
accuracy
number of correctly classified samples divided by all classified samples

Note 1 to entry: It is calculated as $a = (T_\mathrm{P} + T_\mathrm{N}) / (T_\mathrm{P} + F_\mathrm{P} + T_\mathrm{N} + F_\mathrm{N})$.

3.2.7
confusion matrix
matrix used to record the number of correct and incorrect *classifications* (3.1.1) of samples

3.2.8
F_1 score
F-score
F-measure
F_1-measure
harmonic mean of *precision* (3.2.9) and *recall* (3.2.10)

Note 1 to entry: It is calculated as $F_1 = 2T_\mathrm{P} / (2T_\mathrm{P} + F_\mathrm{P} + F_\mathrm{N})$.

3.2.9
precision
positive predictive value
number of samples correctly classified as positive divided by all samples classified as positive

Note 1 to entry: It is calculated as $p = T_\mathrm{P} / (T_\mathrm{P} + F_\mathrm{P})$.

3.2.10
recall
true positive rate
sensitivity
hit rate
number of samples correctly classified as positive divided by all positive samples

Note 1 to entry: It is calculated as $r = T_\mathrm{P} / (T_\mathrm{P} + F_\mathrm{N})$.

3.2.11
specificity
selectivity
true negative rate
number of samples correctly classified as negative divided by all negative samples

Note 1 to entry: It is calculated as $s = T_N / (T_N + F_P)$.

3.2.12
false positive rate
fall-out
number of samples incorrectly classified as positive divided by all negative samples

Note 1 to entry: It is calculated as $F_{P,R} = F_P / (F_P + T_N)$.

3.2.13
cumulative response curve
gain chart
graphical method of displaying *true positive rates* (3.2.10) and percentage of positive prediction in the total data across multiple thresholds

3.2.14
lift curve
graphical method of displaying on the y-axis the ratio of *true positive rate* (3.2.10) between the model and a random classifier, and on the x-axis the percentage of positive predictions in the total data across multiple thresholds

3.2.15
precision recall curve
PRC
graphical method for displaying *recall* (3.2.10) and *precision* (3.2.9) across multiple thresholds

Note 1 to entry: A PRC is more suitable than a ROC (receiver operating characteristic) curve for showing performance with imbalanced data.

3.2.16
receiver operating characteristic curve
ROC curve
graphical method for displaying *true positive rate* (3.2.10) and *false positive rate* (3.2.12) across multiple thresholds

3.2.17
cross-validation
method to estimate the performance of a machine learning method using a single dataset

Note 1 to entry: Cross-validation is typically used for validating design choices before training the final model.

3.2.18
majority class
class with the most samples in a dataset

4 Abbreviated terms

AI	artificial intelligence
ANOVA	analysis of variance
AUPRC	area under the precision recall curve

AUROC	area under the receiver operating characteristic curve
CLT	central limit theorem
CPU	central processing unit
CRC	cumulative response curve
FC	fully connected
FDR	false discovery rate
IoU	intersection over union
GPU	graphics processing unit
ROC	receiver operating characteristic

5 General principles

5.1 Generalized process for machine learning classification performance assessment

A generalized process for machine learning classification performance assessment is shown in Figure 1.

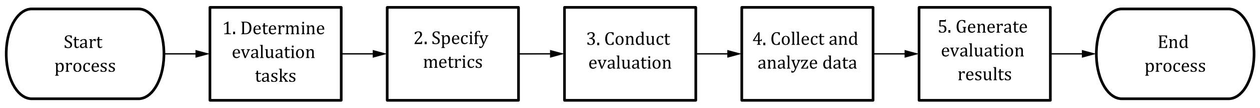

Start process
1. Determine evaluation tasks
2. Specify metrics
3. Conduct evaluation
4. Collect and analyze data
5. Generate evaluation results
End process

Figure 1 — Generalized process for machine learning classification performance assessment

Step 1: Determine evaluation tasks

Determine the appropriate classification task or tasks for the evaluation.

Step 2: Specify metrics

Based on the classification task, specify the required metric or metrics.

Step 3: Conduct evaluation

Create the evaluation plan, implement the evaluation environment including software and hardware, prepare datasets and process datasets.

Step 4: Collect and analyse data

According to the specified metrics, collect model outputs such as classification predictions for each sample.

Step 5: Generate evaluation results

Generate evaluation results based on specified metrics and other relevant information.

5.2 Purpose of machine learning classification performance assessment

The purpose of the assessment and its baseline requirements can vary greatly depending on whether it applies to the "design and development" or "verification and validation” stage.

The purpose of assessment during the “design and development” stage is to optimize hyperparameters to achieve the best classification performance. The purpose of assessment during the "verification and validation" stage is to estimate the trained model performance.

Performance assessment can be applied for several purposes, including:

— model assessment, to know how good the model is, how reliable the model's predictions are, or the expected frequency and size of errors;

— model comparison, to compare two or more models in order to choose between them;

— out-of-sample and out-of-time comparisons, to check that performance has not degraded with new production data.

5.3 Control criteria in machine learning classification performance assessment

5.3.1 General

When assessing machine learning classification performance, consistent approaches and methods should be applied to demonstrate relevance, legitimacy and extensibility. Special care should be taken in comparative assessments of multiple machine learning classification models, algorithms or systems to ensure that no approach is favoured over another.

5.3.2 Data representativeness and bias

Except when done for specific goal-relevant reasons, the training and test data should be as free of sampling bias as possible. That is, the distribution of features and classes in the training data should be matched to their distribution in the real world to the extent possible. The training data does not need to match the eventual use case exactly. For example, in the case of self-driving cars, it can be acceptable to assess the classification performance of machine learning models trained on closed-circuit tracks rather than on open roads for prototype systems. The data used to test a machine learning model should be representative of the intended use of the system.

Data can be skewed, incomplete, outdated, disproportionate or have embedded historical biases. Such unwanted biases can propagate biases present in the training data and are detrimental to model training. If the machine learning operating environment is complex and nuanced, limited training data will not necessarily reflect the full range of input data. Moreover, training data for a particular task is not necessarily extensible to different tasks. Extra care should be taken when splitting unbalanced data into training and test to ensure that similar distributions are maintained between training data, validation data and test data.

Data capture bias can be based on both the collection device and the collector's preferences. Label biases can occur if categories are poorly defined (e.g. similar images can be annotated with different labels while, due to in-class variability, the same labels can be assigned to visually different images). For more information on bias in AI systems, see ISO/IEC TR 24027[1].

5.3.3 Preprocessing

Special care should be taken in preprocessing and its impact on performance assessment, especially in the case of comparative assessment. Depending on the purpose of the evaluation, inconsistent preprocessing can lead to biased interpretation of the results. In particular, when preprocessing favours one model over another, their performance gap should not be attributed to the downstream algorithms. Examples of preprocessing include removal of outliers, resolving incomplete data or filtering out noise.

5.3.4 Training data

Special care should be taken in the choice of training and validation data and how the choice impacts performance assessment, especially in the case of comparative assessment. Depending on the purpose of the evaluation, the use of different training data can lead to a biased interpretation of the results. In particular, in such cases any performance gap should be attributed to the combination of the algorithm and training data, rather than to just the algorithm.

In the context of model comparison, the training data used to build the respective models can differ. One can take two models, trained on different training data, and evaluate them against each other on the same test data.

5.3.5 Test and validation data

The data used to test a machine learning model shall be the same for all machine learning models being compared. The test and validation data shall contain no samples that overlap with training data.

5.3.6 Cross-validation

Cross-validation is a method to estimate the performance of a machine learning method using a single dataset.

The dataset is divided into k segments, where one segment is used for test while the rest is used for training. This process is repeated k times, each time using another segment as the test set. When k is equal to N, the size as the dataset, this is called leave-one-out cross-validation. When k is smaller than N, this is called k-fold cross-validation.

It can be of interest to compare the performance of different cross-validation techniques when all other variables are controlled. However, models whose performance is being compared should not use different cross-validation techniques (e.g. it is not appropriate to compare Model A k-fold cross-validation results against the mean of Model B single train-test split results).

The primary use of cross-validation is for validating design choices such as hyperparameter values, by comparing their overall effect on various models. That is why it is typical to retrain a model on the full dataset after that validation, using the hyperparameters that performed best on average. However, cross-validation does not provide a performance assessment of that final model, and extrapolating performance from the output of cross-validation is a rough approximation with no guarantee of faithfulness.

Another use of cross-validation is for comparative evaluation of machine learning algorithms, without subsequently training a final model. An algorithm is considered to outperform another if on average its resulting models perform best.

5.3.7 Limiting information leakage

Information leakage occurs when a machine learning algorithm uses information not in the training data to create a machine learning model.

Information leakage is often caused when training data includes information not available during production. In an evaluation, information leakage can result in a machine learning model's classification accuracy being overstated. A model trained under these conditions will typically not generalize well.

Evaluations should be designed to prevent information leakage between training and test data.

EXAMPLE A machine learning model can be designed to classify between native and non-native Spanish speakers, using multiple audio samples from each subject. Some observation features, such as vowel enunciation, are potentially useful for this type of speaker classification. However, such features can also be used to identify the specific speaker. The model can use identity-based information to accurately classify test data, even though this information would not be available in production systems. The solution would be to not include the same subject in both training and test data, even if the training and test samples differ.

5.3.8 Limiting channel effects

A channel effect is a characteristic of data that reflects how data were collected as opposed to what data were collected. Channel effects can cause machine learning classification algorithms to learn irrelevant characteristics from training data as opposed to relevant content, which in turn can lead to poor machine learning classification performance.

Channel effects can be caused by the mechanism used to acquire data, preprocessing applied to data, the identity of the individual obtaining data, and environmental conditions under which data were acquired, among other factors.

The data should be as free of channel effects as possible. Controlling channel effects in training data contributes to better performance. Controlling channel effects in test data enables higher-quality assessments.

NOTE One method of reducing channel effects is to balance channel distributions for each class in the data.

Reporting should describe known channel effects introduced to the training data. Channel effects should be accounted for during statistical significance testing (see Clause 7).

EXAMPLE A vision-based system can be designed to distinguish between images of cats and dogs. However, if all “cat” images are high-resolution, and all “dog” images are low-resolution, a machine learning classifier can learn to classify images based on resolution as opposed to content.

5.3.9 Ground truth

Ground truth is the value of the target variable for a particular item of labelled input data. Cleanliness in ground truth can affect classification performance measurement. When assessing classification performance, a strong generalizable ground truth should be established.

General agreement on an aggregated ground truth can be quantified using measurements of agreement such as Cohen's kappa coefficient.

In some domains (e.g. medical), inter-annotator variation can be significant, especially in tasks where team-based consensus is involved.

5.3.10 Machine learning algorithms, hyperparameters and parameters

Most machine learning algorithms have characteristics that affect their learning processes, known as hyperparameters. Machine learning algorithms use hyperparameters and training data to establish internal parameters. The manner in which these parameters are computed can vary. For example, generative algorithms can optimize parameters such that the probability of the available training data is maximized, whereas discriminative algorithms can optimize parameters to maximize classification accuracy.

Hyperparameter types should be reported for all machine learning algorithms in an assessment, as well as hyperparameter values for each machine learning model.

Hyperparameter selection bias should be taken into account when machine learning models are compared. Different machine learning algorithms can have different numbers of hyperparameters with different adjustment capabilities. The degree of overfitting in the training process can then differ across machine learning algorithms.

This is especially pronounced in deep learning with its many combinations of architectures, activation functions, learning rates and regularization parameters. No information from the test set shall be used when adjusting hyperparameters, as this typically leads to over-optimistic performance estimation. When label information is needed for such tuning, it is typically drawn from a separate set of data, called the validation set, which is disjoint from the test set.

This challenge can be addressed through approaches such as nested cross-validation. In this training process, an outer loop measures prediction performance while an inner loop adjusts the hyperparameters of the individual models. In this fashion, methods can choose optimal settings for building predictive models in the outer loop.

See Annex C for summary information on selected machine learning classification benchmark tests, including model parameters and values associated with performance against various datasets.

5.3.11 Evaluation environment

Evaluation environmental requirements are as follows:

— the evaluation environment shall not be modified while the assessment is in progress;

— hardware and system software shall not be modified during the assessment;

— the same test environment should be considered for the machine learning models under assessment.

The evaluation environment should meet the minimum environmental requirements required by the target machine learning model. This increases the utility of evaluation results for environment-dependent performance dimensions such as processing time and processing cost. When it is infeasible to implement the required minimum environmental requirements with the actual application, an evaluation environment can be designed to simulate an actual application. In such cases, the potential impact on environment-dependent evaluation results should be analysed. For example, results for processing time and processing cost are not necessarily reflective of performance in an actual application.

5.3.12 Acceleration

The same test environment should be used for all machine learning models under evaluation. Any use of acceleration during training or testing shall be reported. Each machine learning method under test should be optimized to utilize acceleration when available and appropriate.

Accelerators can be exposed as specialized hardware, graphics processing units (GPUs), application-specific integrated circuits or a set of instructions built into central processing units (CPUs). Other examples of acceleration include sparsity, pruning and other optimizations focused on improving memory bandwidth. Accelerators can be applied to a simple function (e.g. general matrix multiplication) or a complex function (e.g. a complete ResNET function).

5.3.13 Appropriate baselines

A baseline method can be necessary as a basis of comparison for machine learning classification performance. Trivial baselines, such as those that always predict the majority class, are useful to consider for the sake of calibrating metric interpretation, but they should not be the only point of comparison.

5.3.14 Machine learning classification performance context

It is important to consider the overall system (including components and sub-systems) in which a machine learning model will be deployed when assessing machine learning model performance. Further, context such as environmental variables can guide machine learning model classification performance trade-offs. It can be necessary to measure performance with multiple contextual datasets to come to a final conclusion on machine learning model performance.

6 Statistical measures of performance

6.1 General

Machine learning classification can be categorized as binary, multi-class or multi-label. Performance measurement for each category is described in this clause.

The classes are typically from a discrete and unordered set, such that the problem cannot be formalized as a regression task. For example, a medical diagnosis of a set of symptoms can be {stroke, drug overdose, seizure}, there is no order to the class values, and there is no continuous change from one class to another.

6.2 Base elements for metric computation

6.2.1 General

Several classification metrics are derived from the following elements:

— true positive (T_P);

— true negative (T_N);

— false positive (F_P);

— false negative (F_N).

6.2.2 Confusion matrix

A confusion matrix in general will have true classes in columns and predicted classes in rows. While confusion matrices are used to generate several widely applicable classification performance metrics, they can also be used to calculate metrics for specialized applications. See Annex D for an example of one such metric, cause-specific mortality fraction (CSMF).

6.2.3 Accuracy

Accuracy should not be used to express comparative performance across models unless classes are known to be reasonably balanced.

6.2.4 Precision, recall and specificity

As precision increases, more true positives are detected, but false negatives are not accounted for. Precision of a class is calculated as:

$$p = \frac{T_P}{T_P + F_P}$$

As recall increases, more true positives are detected, but false positives are not accounted for. Recall of a class is calculated as:

$$r = \frac{T_P}{T_P + F_N}$$

As specificity increases, more true negatives are detected, but false negatives are not accounted for. Specificity of a class is calculated as:

$$s = \frac{T_N}{T_N + F_P}$$

6.2.5 F_1 score

F_1 score is calculated as:

$$F_1 = \frac{2T_P}{2T_P + F_P + F_N}$$

6.2.6 F_β

Precision and recall are not necessarily of equal importance in an AI application, such as when the cost of a false positive is much higher than the cost of a false negative. The F_1 score value can be varied to account for these cases, expressed as F_β[2].

β values greater than 1 indicate that recall is more important than precision, meaning that false negatives are to be minimized. β values lower than 1 indicate that precision is more important than recall, meaning that false positives are to be minimized. F_β is calculated as:

$$F_\beta = \left(1+\beta^2\right)\frac{p*r}{r+\beta^2*p}$$

where

p	is precision of a class;
β	is a factor indicating how much more important recall is than precision;
r	is recall of a class.

The precision and recall weight can also be directly applied with the pair value of (α , β) through the following:

$$F(\alpha p, \beta r) = \frac{(\alpha+\beta)p*r}{\alpha r+\beta p}$$

6.2.7 Kullback-Leibler divergence

Kullback-Leibler Divergence (D_{KL}) is a well-known measure for quantifying the difference between a target distribution and an estimated distribution. While often used as a loss function, it is also used as an evaluation metric over a dataset. In such cases the considered distribution is about the prevalence of each label in the dataset, rather than the probability distribution of each label for an individual sample.

A general formula over all classes *x* is as follows:

$$D_{\mathrm{KL}}\ (p,q) = \sum p(x)\log\left(\frac{p(x)}{q(x)}\right)$$

where

p(x)	is the ground truth or target distribution;
q(x)	is the estimated distribution from the classifier.

6.3 Binary classification

6.3.1 General

In binary classification, each sample is labelled as one of two mutually exclusive classes. These classes are often “positive” or “negative” with reference to a categorization. The two classes are typically unordered.

EXAMPLE Machine learning classification software learns to mark email as “spam” or “not spam” based on input provided by the email recipient.

One-class (also known as unary) classification is similar to binary classification. In unary classification, a single target class exists, along with an outlier class comprised of anything not in the target class. In contrast to binary classification, this outlier class is not explicitly modelled. One-class classifiers are typically used when training data is highly imbalanced, such as when training data only exists for a single target class. In this way, the data from a single class is used to build a model.

6.3.2 Confusion matrix for binary classification

In binary classification, one class is considered as positive and the other one negative, so drawing the confusion matrix is equivalent to computing true and false positives and negatives. Table 1 shows the form of a confusion matrix for a binary classifier.

Table 1 — Elements of a binary confusion matrix

		true classes	
		positive	negative
predicted classes	positive	true positive (T_P)	false positive (F_P)
	negative	false negative (F_N)	true negative (T_N)

6.3.3 Accuracy for binary classification

In the case of binary classification, the application of the definition of accuracy leads to the following computation:

$$a = \frac{(T_P + T_N)}{(T_P + F_P + T_N + F_N)}$$

6.3.4 Precision, recall, specificity, F_1 score and F_β for binary classification

In the case of binary classification, the terms precision, recall, specificity, F_1 score and F_β refer to the computation of those metrics for the positive class.

6.3.5 Kullback-Leibler divergence for binary classification

For a binary classifier, the formula for Kullback-Leibler divergence is as follows:

$$D_{KL} = \frac{(T_P + F_N)*\log\frac{(T_P + F_N)}{(T_P + F_P)} + (T_N + F_P)*\log\frac{(T_N + F_P)}{(T_N + F_N)}}{N}$$

Where N is the number of samples.

6.3.6 Receiver operating characteristic curve and area under the receiver operating characteristic curve

A ROC curve is a graphical method for displaying true positive rates and false positive rates across multiple thresholds from a binary classifier.

To express performance across all thresholds, the area under the receiver operating characteristic curve (AUROC) can be calculated. A higher AUROC indicates more robust performance, ranging from 0 (worst) to 1 (best). Classifiers that perform no better than chance will have an AUROC of 0,5.

AUROC is well-suited for cases where ranked predictions are important. It is also well-suited for cases where false positive rates and true positive rates are of roughly equal importance. AUROC is not well-suited for cases in which data are imbalanced, because it does not account for the proportion of false positives and true positives.

See Annex B for an illustration of a ROC curve derived from binary classification outputs.

6.3.7 Precision recall curve and area under the precision recall curve

Precision recall curve (PRC) is a graphical method for displaying recall and precision across multiple thresholds. The graph plots recall on the x-axis against precision on the y-axis.

To express performance across all thresholds, the area under the PR curve (AUPRC) can be calculated. A higher AUPRC indicates more robust performance. The closer the PRC is to the upper right corner, the better the classifier.

AUPRC is well-suited for cases where achieving good results on the positive class is important, as well as when data are imbalanced.

6.3.8 Cumulative response curve

Cumulative response curve (CRC), also referred as gain curve or gain chart, is a graphical method of displaying true positive rates and percentage of positive predictions in the total data across multiple thresholds.

To express performance across all thresholds, the area under the cumulative response curve can be calculated. Higher values indicate more robust performance, ranging from 0 (worst) to 1 (best). Classifiers that perform no better than chance will have a value of 0,5.

The CRC provides an alternative to ROC. The CRC is useful in cases where the goal is to target a certain proportion of the dataset.

6.3.9 Lift curve

Lift curve is a graphical method of displaying on the y-axis the ratio of true positive rate between the model and a random classifier, and on the x-axis the percentage of positive predictions in the total data across multiple thresholds. Hence the lift curve is directly related to the gain chart.

The lift curve shows the advantage provided by the classifier over random guessing at different percentages of the total data.

6.4 Multi-class classification

6.4.1 General

In multi-class classification, each sample is labelled as one of three or more mutually exclusive classes.

EXAMPLE Machine learning classification software learns to categorize images as “dog”, “cat” or “other” based on labels assigned by a human reviewer.

6.4.2 Accuracy for multi-class classification

Accuracy is a typical evaluation metric for a multi-class classifier. In that case it corresponds to the sum of true positives for each class, divided by the total number of elements in all classes. Per-class accuracy can also be computed in the multi-class case, but it is equivalent to the recall of that class.

6.4.3 Macro-average, weighted-average and micro-average

Several multi-class classification metrics are based on the averaging of per-class metrics: precision, recall, specificity and F_1 score. Multi-class performance can be expressed using one or more of macro-average, weighted-average and micro-average approaches. A basis for selection of the appropriate multi-class performance approach shall be reported.

The two-class concepts of positive and negative can be generalized for a multi-class problem by considering samples in the target class to be positive and samples in all other classes to be negative. For example, false positives with respect to class *i* are samples belonging to other classes that are incorrectly classified as class *i*.

Using F_1 for illustration, macro-average F_1, weighted-average F_1 and micro-average F_1 approaches are as follows:

— Macro-average F_1 averages F_1 for each class without accounting for the number of samples within each class.

— Weighted-average F_1 acknowledges class imbalance, weighting the F_1 score of each class as a function of class size. It is calculated by multiplying the F_1 score of each class by the number of class samples, then dividing by total samples.

— Micro-average F_1 aggregates precision and recall across all classes as one monolithic set. It is calculated by summing true and false positives and negatives for all classes then calculating F_1 from these aggregates.

The three approaches target different use cases. Macro-average F_1 (M_{AC,F_1}) is appropriate when good performance is equally important on each class, regardless of their prevalence.

Micro-average F_1 (M_{IC,F_1}) is more relevant when achieving good performance on a given sample is equally important, regardless of its class.

Weighted-average F_1 (W_{TD,F_1}) favours the performance on the majority class, so that it typically reinforces class imbalance compared to micro-average. Its use is more appropriate when priority is set on handling that majority class rather than minority classes.

Formulae for multi-class machine learning classification performance approaches are as follows:

Let $T_{\mathrm{P}i}$ denote the number of samples correctly classified as class *i.*

Let $F_{\mathrm{P}i}$ denote the number of samples from other classes incorrectly classified as class *i.*

Let $F_{\mathrm{N}i}$ denote the number of samples from class *i* incorrectly classified as another class.

Let *L* denote the total number of classes.

Let *N* denote the total number of samples in all classes.

$$M_{\mathrm{AC},F_1} = \frac{1}{L}\sum_{i=1}^{L}\frac{2*T_{\mathrm{P}i}}{2*T_{\mathrm{P}i}+F_{\mathrm{P}i}+F_{\mathrm{N}i}}$$

$$W_{\mathrm{TD},F_1} = \sum_{i=1}^{L}\frac{T_{\mathrm{P}i}+F_{\mathrm{N}i}}{N} * \frac{2*T_{\mathrm{P}i}}{2*T_{\mathrm{P}i}+F_{\mathrm{P}i}+F_{\mathrm{N}i}}$$

$$M_{\mathrm{IC},F_1} = \frac{\sum_{i=1}^{L}2*T_{\mathrm{P}i}}{\sum_{i=1}^{L}2*T_{\mathrm{P}i}+F_{\mathrm{P}i}+F_{\mathrm{N}i}}$$

The same approach can be used to calculate micro-average, macro-average and weighted-average results for precision, recall and specificity. However, for those metrics the most typical approach is macro-average, as most of their other variants have equivalence properties with other existing metrics.

See Annex A for an illustration of the progression from a confusion matrix to multi-class performance reporting.

6.4.4 Distribution difference or distance metrics

Another way to assess multi-class machine learning classification performance evaluation considers the difference in class distributions between labelled data and predicted data, using the following:

Let *T* be the total number of samples.

Let T_i be the number of samples for each class.

Let P_i be the number of samples predicted for each class.

Construct discrete distributions $T_\mathrm{d} = (T_1, T_2, \dots, T_n)/T$ and $P_d = (P_1, P_2, \dots P_n)/T$.

Compute the difference in the distributions.

For Kullback-Leibler divergence, the difference metric is computed as

$$D_\mathrm{KL} = \Sigma\ (P_i) * l_\mathrm{n} \left({}^{P_i}\!/_{T_i} \right)$$

NOTE 0 * *log* (0/*x*) == 0.

6.5 Multi-label classification

6.5.1 General

In multi-label machine learning classification, a sample can be labelled as one or more classes. Classes are not mutually exclusive, and a sample can have multiple labels. Further, classes can be correlated.

Multi-label machine learning classification performance assessment is complicated by the fact that a sample can have multiple labels. A machine learning model can predict a subset of all correct labels for a given sample. Alternatively, a machine learning model can correctly predict some labels but incorrectly predict others for a given sample.

EXAMPLE Multi-label machine learning classification software learns to categorize text as one or more of opinion, news, hostile, sympathetic, misinformation or disinformation based on labels assigned by a human reviewer. A given text can be labelled as hostile, opinion and disinformation. A different text can be labelled news.

Different metrics are available to assess multi-label machine learning classification performance, including Hamming loss, exact match ratio and the Jaccard index. A number of metrics for binary and multi-class classification are also applicable, often after some adaptations. For instance, exact match ratio is an adaptation of binary accuracy.

A basis for selection of the metric for multi-label performance assessment shall be reported.

6.5.2 Hamming loss

Hamming loss is the number of incorrectly predicted labels divided by the total number of labels. Lower values represent more robust performance. Hamming loss treats positive and negative errors equally and is therefore useful as a general measure of performance.

Hamming loss is calculated as follows:

Let *N*, *L* denote the total number of samples and labels present in a dataset.

Let $\hat{l}_{i,j} = \hat{l}_{i,1}, \hat{l}_{i,2}, \dots, \hat{l}_{i,\mathrm{L}}$ denote the predicted label values for x_i. $\hat{l}_{i,j} = 1$ if label *j* is present among the predicted labels for x_i, otherwise $\hat{l}_{\mathrm{i,j}} = 0$.

Similarly, let $l_{i,j} = l_{i,1}, l_{\mathrm{i},2}, \dots, l_{i,\mathrm{L}}$ denote the ground truth label values for x_i. $l_{i,j} = 1$ if label *j* is present among the ground truth labels for x_i, otherwise $l_{\mathrm{i,j}} = 0$.

Hamming loss for data sample x_i can be expressed as follows:

$$H_\mathrm{L}\left(x_i\right) = \frac{1}{L}\sum_{j=1}^{L} I\left(\hat{l}_{i,j} \neq l_{i,j}\right)$$

Total Hamming loss is the average Hamming loss over all data samples. Total Hamming loss can be expressed as follows:

$$H_\mathrm{L} = \frac{1}{N\,L}\sum_{i=1}^{N}\sum_{j=1}^{L} I\left(\hat{l}_{i,j} \neq l_{i,j}\right)$$

6.5.3 Exact match ratio

Exact match ratio, or subset accuracy, is the percentage of samples for which all labels are predicted and predicted accurately. Given a sample with correct labels *A*, *B* and *C*, exact match ratio treats a prediction with labels *A* and *B* (but not *C*) as an error. This approach treats partially correct predictions as errors. This coarse approach can be appropriate for first-order assessment when data are highly imbalanced.

Exact match ratio is calculated as follows:

Let N, L denote the total number of samples and labels present in a dataset.

Let I denote the indicator function.

Let $\hat{l}_i$ and l_i denote the set of predicted and ground truth label values for data sample x_i respectively.

The exact match ratio can be expressed as follows:

$$E_\mathrm{MR}\ \frac{1}{N}\sum_{i=1}^{N} I\left(\hat{l}_i = l_i\right)$$

6.5.4 Jaccard index

Another frequently used metric for multi-label machine learning classification is the Jaccard index, commonly referred to as intersection over union (IoU).

I_oU is calculated as follows:

Let *P* denote sets of predicted labels.

Let *T* denote sets of true labels.

$$I_\mathrm{oU} = \frac{\left|T \cap P\right|}{\left|T \cup P\right|}$$

This metric can be applied at the dataset level (by aggregating all labels from all samples) or at the sample level (by computing sample-specific IoUs and averaging them).

6.5.5 Distribution difference or distance metrics

Similar to the distribution difference method to assess multi-class machine learning classification performance, multi-label machine learning classification evaluation considers the difference in class distributions between labelled data and predicted data, using the following:

Let T_i be the number of samples for each class. Samples can belong to multiple classes.

Let T_t be the number of total labels assumed by all samples.

Let P_i be the number of samples predicted for each class.

Let T_p be the number of total labels assumed by all predictions.

Construct discrete distributions $T_\mathrm{d} = (T_1, T_2, \ldots, T_n)/T_t$ and $P_\mathrm{d} = (P_1, P_2, \ldots, P_n)/T_p$.

Compute the difference in the distributions.

For Kullback-Leibler divergence, the difference metric is computed as

$$D_\mathrm{KL} = \Sigma\ (P_i)^* l_n \left({}^{P_i}\!\!\diagup\!_{T_i} \right)$$

NOTE 0 * log (0/x) == 0.

6.6 Computational complexity

6.6.1 General

Additional machine learning classification performance attributes can be taken into consideration with respect to the implementation and deployment of machine learning classification systems. In addition to accuracy metrics derived from confusion matrixes, assessors can consider latency, throughput, efficiency and energy consumption as elements of a four-dimensional optimization space when assessing the classification performance of a machine learning system. One or more of these elements can constrain joint optimization across these dimensions when developing or assessing a system.

6.6.2 Classification latency

For interactive, user-facing classification applications, it can be necessary to estimate classification latency. The duration from user input to machine learning model inference can determine the quality of service provided. This duration should fall within latency bounds that define the operational constraints of the implementation.

EXAMPLE Hyperparameter optimization can be used to improve model accuracy, potentially with increased model complexity. In some cases, a resultant optimized model that fails to meet latency thresholds is considered unacceptable.

Classification latencies can be caused by factors such as slow feature storage, larger batch sizes that trade off latency with throughput, model complexity and inefficient resource tuning.

To serve classification models, specialized hardware and accelerators can be used in an end-to-end pipeline to meet latency constraints, thereby leading to large configuration space. Queuing delays in a heterogeneous pipelined arrangement can emerge due to variable speeds and their association with the interarrival process for a given system configuration.

Classification latency can be expressed as follows:

$$\frac{1}{N}\sum_{i=1}^{N}(T_\mathrm{mr} - T_\mathrm{di})$$

where

N	is the total number of samples in a given dataset;
T_mr	is the time at which the machine learning model generates an inference for the i^{th} sample;
T_di	is the time at which the i^{th} sample is ingested into the inference processing pipeline.

6.6.3 Classification throughput

Classification throughput is the number of inferences per unit of time that a machine learning model computes, given latency constraints.

Classification throughput can be expressed as follows:

$$\frac{N_{\text{cla}}}{T_{\text{e}} - T_{\text{b}}}$$

where

N_{cla}	is the total number of samples for which a machine learning model generates an inference in a given period of time under latency constraints;
T_{e}	is the time at which inference ends for the N_{cla} samples;
T_{b}	is the time at which begins for the N_{cla} samples.

NOTE Latency constraints can be unbounded if specified by user or system designer. For example, the throughput of an AI device or AI system can be improved by performing classification in a grouped or batched inference manner. This enables greater utilization of the computational units on the system and increased reuse of filter weights in inference operations, especially on multi-threaded and larger compute devices. However, grouped or batched operations can result in higher latencies for each of the individual classifications.

6.6.4 Classification efficiency

Classification efficiency is the degree to which an objective (considered an input-to-output ratio) is achieved economically. The objective is a function of accuracy attributes as discussed in 6.2 to 6.5.

Classification efficiency is an important measure of advancement in machine learning. Efficient classifiers require fewer computations to train a particular function than inefficient classifiers.

EXAMPLE Since 2012, the amount of computation needed to train neural networks to achieve the same performance in the ImageNet classification has decreased by a factor of two every 16 months[3].

6.6.5 Energy consumption

It is important to consider how performance and implementation of an AI system will be constrained by the energy consumption bounds (such as performance per watt). This issue is particularly important when an AI system is extensively used or widely adopted. This topic is considered in References [4],[5],[6],[7],[8],[9] and [10].

For many use cases, satisfactory performance at lower energy cost can be desired or practical over the best performance possible. In many cases this allows for trade-off of power and performance.

This trade-off, performance per power (watt), can be computed by measuring total performance P for a given interval T. Average performance can be represented as P/T. Similarly, average power P for a given interval T can be represented by E/T, where E is energy measured in joules.

Therefore, performance per watt can be represented by $(P/T)/(E/T) = P/E$.

For example, in the use case of visual inference, if performance is measured in number of frames classified per second (F/t), then the performance-per-watt becomes frames-per-joule (F_{PJ}). A corresponding measure, Joules per frame (J_{PF}), can be represented as follows:

$$J_{\mathrm{PF}} = \frac{\int_{t=0}^{t=T} P(t)\,dt}{F}$$

Where F is the total number of frames inferred and T is the total time required to analyse F frames. In other cases, it is more desirable to analyse J_{PF} for accurately classified frames.

A more generalized form for different use cases can be represented as follows:

$$J_{\mathrm{PI}} = \frac{\int_{t=0}^{t=T} P(t)\,dt}{I}$$

Where J_{PI} is Joules per inference and I represents the number of accurately classified inferences.

NOTE Higher energy consumption increases the costs of an AI system.

EXAMPLE In a data centre application, careful consideration of energy consumption can be necessary because the deployment of the AI solution is constrained by available cooling and power delivery.

While these energy consumption considerations occur at the point of inference, energy consumption during training is also relevant, as discussed in References [11] and [12].

7 Statistical tests of significance

7.1 General

Differences in accuracy and other measures of machine learning classification performance can depend greatly on the particular data on which those measures were evaluated. It is therefore necessary to account for the test data when making claims about machine learning classification performance. This accounting shall include the number and distribution of samples available for evaluation.

This issue is of particular importance in evaluations where a relatively small dataset is used for testing. It is also important if the evaluation methodology is based on approaches such as k-fold cross-validation that use multiple permutations of a dataset to train and test a model. In different ways, the techniques discussed in this clause attempt to address whether differences in model performance are based on chance.

The scientific community brings varied approaches to this area, e.g. with regards to natural language processing. The statistical tests of significance discussed in this clause are among the primary methods used in practice. However, many additional tests exist for different scenarios and applications, and the question of which tests are optimal for different scenarios and applications is not settled.

Evaluations should report on the application of statistical tests of significance to machine learning classification performance results. If no statistical tests of significance were performed or no analysis was applied, evaluations should report accordingly.

7.2 Paired Student's t-test

A paired Student‘s t-test compares the means and standard deviations of two groups to determine if differences between the groups are significant. The paired Student's t-test assumes normal distribution, and it is not applicable when three or more groups are being evaluated.

For machine learning classification, this speaks to the question as to whether differences in accuracy between models are statistically significant.

While the paired Student's t-test is commonly used by practitioners, it should not be used in evaluations based on k-fold approaches. Data resampling across multiple training sets means that values are no longer independent, contradicting an underlying assumption of this type of test.

A more robust approach is to use 5x2 cross-validation, as introduced by Dietterich.[13] Designed to mitigate the issue of sample dependence, this test uses five runs of two-fold cross-validation. In each run, data are divided into training and test. Models are both trained and tested on each set, and performance is calculated for each permutation.

7.3 Analysis of variance

When comparing more than two groups, analysis of variance (ANOVA) can be used to determine whether the means of more than two groups are equal (i.e. whether differences in accuracy between three or more models are statistically significant). ANOVA assumes normal distribution and that variance is homogenous.

ANOVA is based on between-group and within-group mean-squared values. The sum of squared differences between groups expresses group means' deviance from the overall mean. The sum of squares within-group is based on the squared sum of values, centred on each group's overall mean. This expresses measurement variance within each group.

The F-statistic is the ANOVA test statistic, calculated as the ratio of the between-group and within-group mean squared values.

7.4 Kruskal-Wallis test

The Kruskal-Wallis test is a non-parametric, rank-based method for testing whether samples originate from the same distribution. It can be used to compare the performance of three or more independent groups. If the Kruskal-Wallis test value is greater than the critical chi-square value, then the groups have a different distribution.

7.5 Chi-squared test

The Chi-squared test is a method for determining for independent, categorical variables whether observed and expected frequencies match. The Chi-squared test uses a contingency table to determine whether two variables are associated, resulting in a test statistic with a chi-squared distribution. Large chi-squared values indicate poorly matched observed and expected frequencies.

7.6 Wilcoxon signed-ranks test

The Wilcoxon signed-ranks test[14] is a non-parametric alternative to the paired Student's t-test applied to ranked data. For each dataset, the test ranks the performance differences for two classifiers and compares ranks for positive and negative differences. The goal is to determine whether any randomly selected observation will be greater or less than a sample in the other dataset's distribution. Ranked values for both datasets are interleaved to identify any clusters at opposite ends, which would indicate that the results fail the significance test.

7.7 Fisher's exact test

Fisher's exact test is a test of statistical significance used in the analysis of contingency tables, matrices that show variables' frequency distribution. It is applicable when analysing two nominal variables (categorical variables with no assumptions of rank or order). Fisher's exact test investigates if a nominal variable's proportions differ from those of another nominal variable, particularly for tests with small sample sizes.

7.8 Central limit theorem

The central limit theorem (CLT) states that as sample size increases, the mean values of all the samples will approximately take the shape of Gaussian distribution around the population mean. Each sample is defined as a group of observations drawn from the same underlying population distribution, which is a part of a larger searchable population. The population is defined as a search space with all possible observations captured during a trial. CLT-based tools can be employed to estimate the likelihood that two samples with different accuracy scores were drawn from the population with different underlying distributions.

By measuring the error distribution for a given model, the CLT method can also explain whether the classification error for a given model can be attributed to noise or to the absence of a critical feature. In other words, if the error distribution for a given model is not normally distributed or is skewed, it can be attributed to a faulty model or missing feature. Alternatively, independent calculations of model accuracy on multiple samples approximate the model skill distribution around overall mean accuracy for a given problem.

7.9 McNemar test

The McNemar test is a non-parametric test applied to paired nominal data represented in contingency tables. It is suitable for use when training cannot feasibly be executed multiple times, precluding approaches such as k-fold cross-validation.

The McNemar test attempts to answer whether the contingency table is homogeneous. It does this by analysing cases where model classification results differ, such as aggregate results where machine learning model *A* classifies a sample as “dog” and machine learning model *B* classifies that same sample as “not a dog”. The test determines whether models’ relative proportion of errors differ. Variants of the McNemar test can be used when values in the contingency table are small.

7.10 Accommodating multiple comparisons

7.10.1 General

The multiple comparisons problem occurs when a set of statistical inferences are considered simultaneously. For example, if a statistical test is performed at the $\alpha = 5\ \%$ level and the corresponding null hypothesis is true, there is only a 5 % chance of incorrectly rejecting the null hypothesis. However, if tests are performed, and all corresponding null hypotheses are true, the expected number of incorrect rejections is 0,05 * *n*. If the tests are statistically independent from each other, the probability of at least one incorrect rejection grows with the number of tests. That is, the family-wise error rate is

$$1-(1-\alpha)^m$$

Where m is the number of tests.

When comparing classifiers, the multiple comparisons problem would exist when a single classifier is compared against several others, individually, or when multiple hyperparameterizations of these classifiers are compared.

7.10.2 Bonferroni correction

The Bonferroni correction is as follows:

Let $H_1, \ldots, H_m$ be a family of m null hypotheses, with $p_1, \ldots, p_m$ as their corresponding p- values.

Order the p-values by increasing value, $p'_1, \ldots, p'_m$, where their associated hypotheses are $H'_1, \ldots, H'_m$.

Given an empirical significance level α, k is the minimal index such that

$$p'_k > \frac{\alpha}{m+1-k}$$

Reject null hypotheses $H'_1, \ldots, H'_{k\text{-}1}$, and do not reject the others.

7.10.3 False discovery rate

False discovery rate (FDR) is an approach to adjust the p-values of each test in a multi-comparison study. FDR describes the rate of which a given set of hypothesis tests would falsely identify a significant test. FDR is less conservative than the Bonferroni approach and has greater ability to find truly significant results.

8 Reporting

The following should be reported:

— source, size and composition of training data;

— source, size and composition of test data;

— efforts taken to analyse, account for and reduce bias in test and training data;

— methods by which ground truth is established in test and training data;

— reliability of ground truth in test and training data and its potential impact on statistical significance;

— number of true and false positive instances correctly and incorrectly classified at representative operating points;

— test environment to include hardware (CPU/GPU or other processing architecture) and software (operating system) used to generate inferences, with specific versions and generations;

— inference generation duration or other measures of computational efficiency.

Annex A
(informative)

Multi-class classification performance illustration

A.1 Progression from raw classification outputs to multi-class results

Tables A.1 to A.4 illustrate the progression from raw classification outputs to class-specific performance to multi-class results, using approaches described in 6.4.2 and 6.4.3. Table A.1 shows a confusion matrix for classes A, B and C. True positive results are shown in bold across the diagonal axis.

Table A.1 — Confusion matrix for multi-class classification with classes A, B, C

		Actual		
		A	**B**	**C**
Predicted	**A**	**400**	150	14
	B	23	**3800**	144
	C	13	355	**65**

Table A.2 shows notional true positive, true negative, false positive and false negative counts for each class. Precision, recall and other metrics can be calculated from these counts.

Table A.2 — Notional true positive, true negative, false positive and false negative counts for classes A, B, C

	A	**B**	**C**
T_P	400	3800	65
T_N	4364	492	4373
F_P	164	167	368
F_N	36	505	158

Table A.3 shows binary accuracy, precision, recall, specificity and F_1 for each class, calculated from the counts shown in Table A.2.

Table A.3 — Accuracy, precision, recall, specificity and F_1 for classes A, B and C (in per cent)

	A	**B**	**C**
Accuracy	91,74	88,27	29,15
Binary Accuracy	95,97	86,46	89,40
Precision	70,92	95,79	15,01
Recall	91,74	88,27	29,15
Specificity	96,38	74,66	92,24
F_1	80,00	91,88	19,82

Table A.3 shows how accuracy computation, expressed as per cent, differs from the application of binary classification metrics. Overall accuracy in the Table A.3 example is 85,92 %

Table A.4 shows micro, macro and weighted multi-class results, calculated from Table A.2 and Table A.3 using the formulae shown in 6.4.2. Table A.4 also shows how accuracy computation differs from the application of binary classification metrics. Because Class B is much larger than Classes A and C, it is a strong determinant of weighted performance.

Table A.4 — Macro, weighted and micro multi-class results (in per cent)

	Macro	**Weighted**	**Micro**
Accuracy	90,61	87,43	90,61
Precision	60,57	89,98	85,92
Recall	69,72	85,92	85,92
Specificity	87,76	77,36	92,96
F_1	63,90	87,60	85,92

Annex B
(informative)

Illustration of ROC curve derived from classification results

B.1 Progression from raw binary classification output to ROC curve

The following figures illustrate the progression from raw binary classification output to ROC curve, as described in 6.3.4. The classification output is first sorted in descending order, as shown in Table B.1.

Table B.1 — Example of classification output for ROC generation

Predicted probability	Actual class
1,00	Yes
0,96	Yes
0,94	Yes
0,86	Yes
...	...
0,03	No
0,03	Yes
0,00	No

At each row in the table, a confusion matrix can be created counting the number of true and false positives and negatives. The T_P and F_P rates at that probability threshold are then plotted in the ROC space. The process is illustrated in Figures B.1 to B.7, where "X" shows an incorrect classification at this threshold and a black circle shows correct classifications:

a) plotting actual class values against predicted probabilities of each sample (Figure B.1);

b) threshold = 0,99 (vertical line), where T_P rate = 0,18 and F_P rate = 0 (Figure B.2);

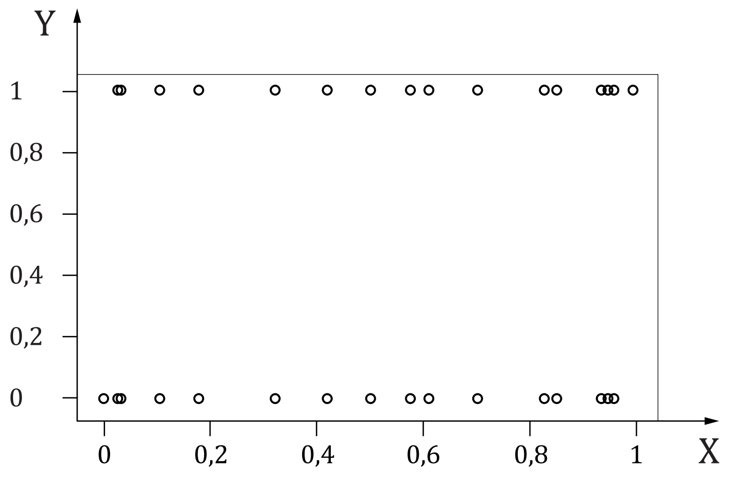
c) threshold = 0,9, where T_P rate = 0,68 and F_P rate = 0,02 (Figure B.3);

d) threshold = 0,7, where T_P rate = 0,75 and F_P rate = 0,03 (Figure B.4);

e) threshold = 0,3, where T_P rate = 0,96 and F_P rate = 0,14 (Figure B.5);

f) threshold = 0,01, where T_P rate = 1 and F_P rate = 0,59 (Figure B.6);

g) ROC curve rendered from performance at multiple operating points y (Figure B.7).

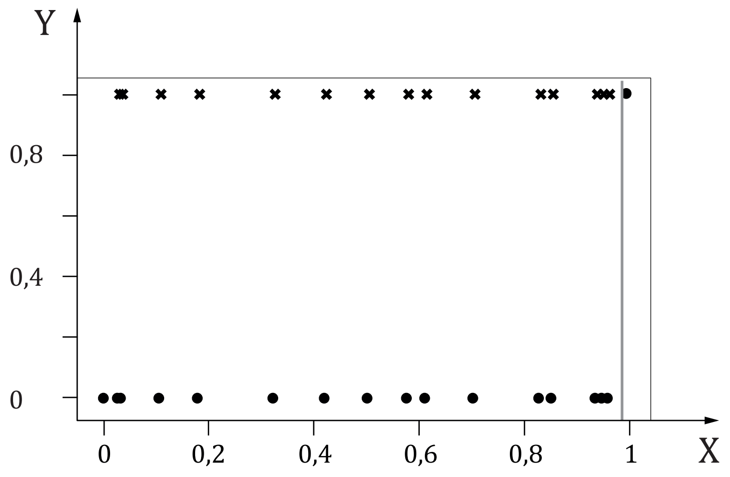


Key

X predicted probability

Y actual values

Figure B.1 — Plotting of actual class values against predicted probabilities of each sample

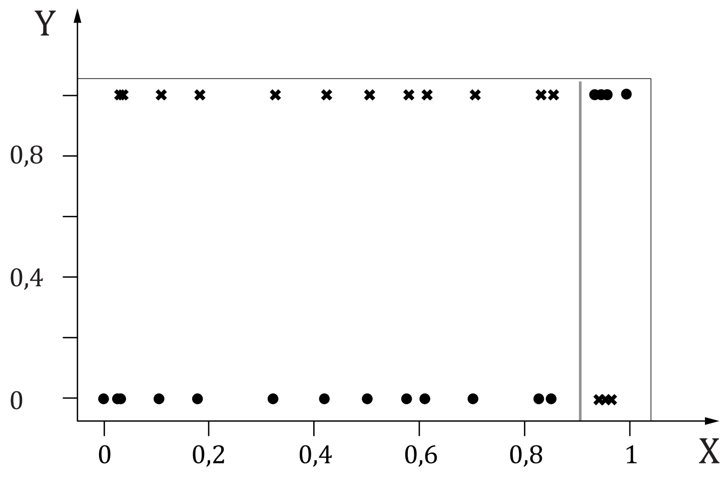


Key

X predicted probability

Y actual values

Figure B.2 — Threshold = 0,99, T_P rate = 0,18, F_P rate = 0

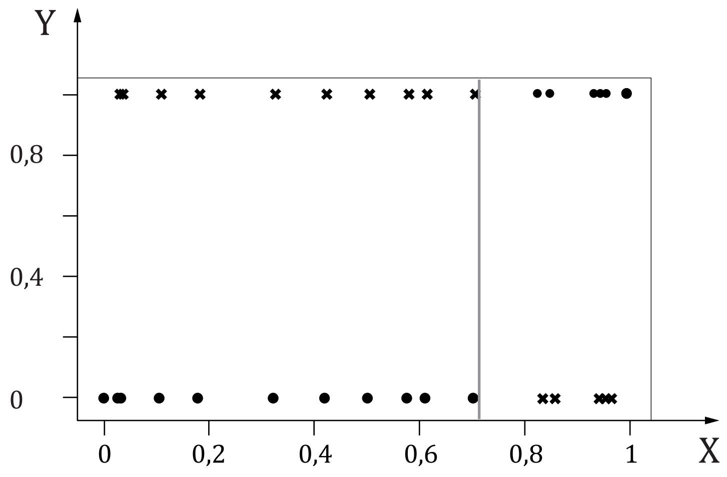


Key

X predicted probability

Y actual values

Figure B.3 — Threshold = 0,9, T_P rate = 0,68, F_P rate = 0,02

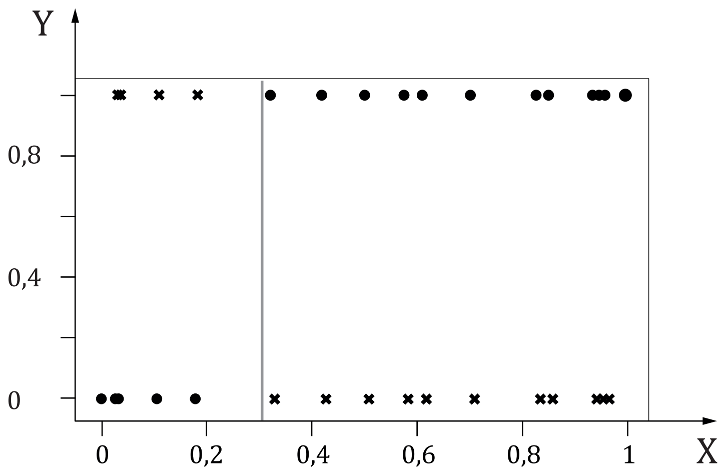


Key

X predicted probability

Y actual values

Figure B.4 — Threshold = 0,7, T_P rate = 0,75, F_P rate = 0,03

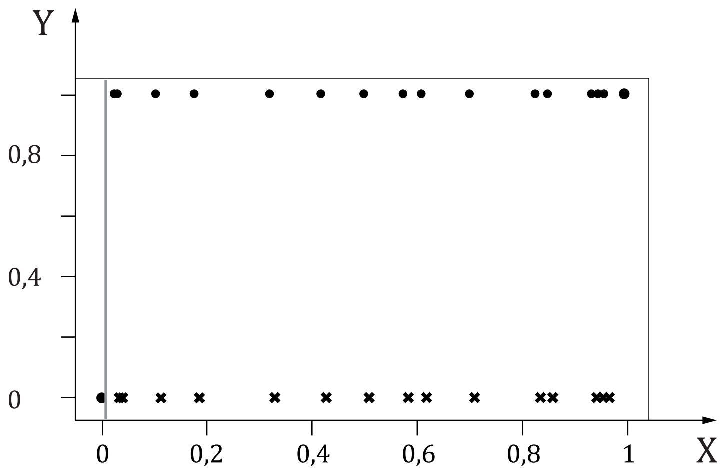


Key

X predicted probability

Y actual values

Figure B.5 — (e) threshold = 0,3, T_P rate = 0,96, F_P rate = 0,14



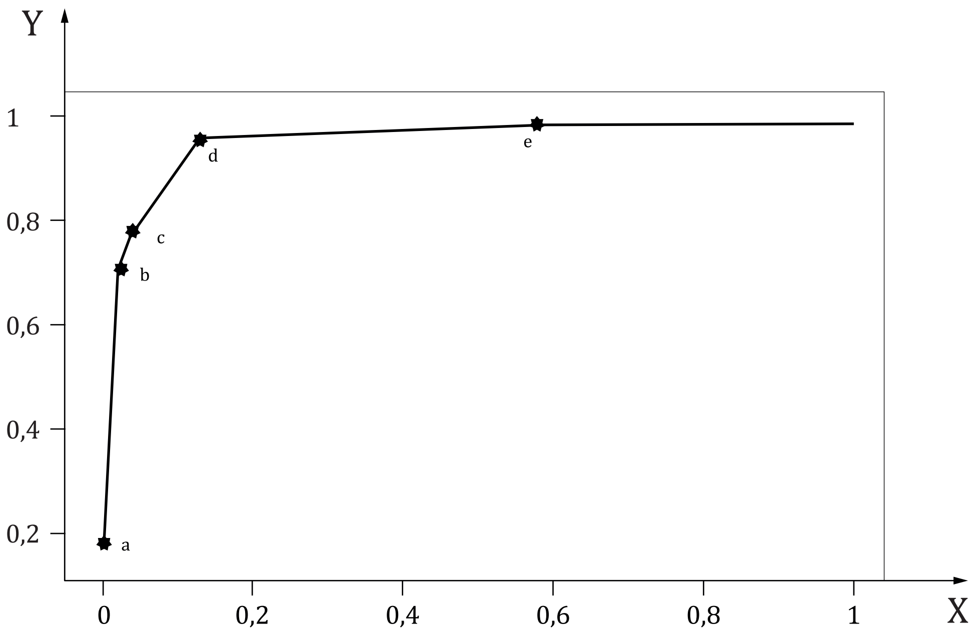
Key

X predicted probability

Y actual values

Figure B.6 — Threshold = 0,01, T_P rate = 1, F_P rate = 0,59

The points are then joined to create the ROC curve:



Key

X	false positive rate
Y	true positive rate
[a]	Threshold = 0,99.
[b]	Threshold = 0,9.
[c]	Threshold = 0,7.
[d]	Threshold = 0,3.
[e]	Threshold = 0,01.

Figure B.7 — ROC curve rendered from performance at multiple operating points *y*

Annex C
(informative)

Summary information on machine learning classification benchmark tests

C.1 Examples of machine learning classification benchmark tests

Table C.1 lists summary information from machine learning classification performance tests, including convolutional (shown as "conv") and fully connected (FC) layers.

Table C.1 — Summary information from machine learning classification benchmark tests

Dataset	Model	Accuracy	Top-1 error	Top-5 error	Model depth	Parameters	Model parameters and values
ImageNet	AlexNet[15]	63,3 %	37,5 %	17,0 %	8 conv: 5 FC: 3	60M	batch size: 128 momentum: 0,9 weight decay: 0,0005 learning rate: 0,01
	VGG-19[16]	74,5 %	25,5 %	8 %	19 conv: 16 FC: 3	144M	batch size: 256 momentum: 0,9 weight decay: 0,000 5 learning rate: 0,01
	ResNet-50[17]	77,15 %	22,85 %	6,7 %	50 conv: 49 FC: 1	20M	batch size: 256 momentum: 0,9 weight decay: 0,0001 learning rate: 0,1
	Efficient-Net-B7[18]	84,4 %	15,6 %	2,9 %	813	66M	momentum: 0,9 weight decay: 0,0001 learning rate: 0,256
MNIST	RMDL[19]	99,82 %					
	MCDNN[20]	99,77 %			5 conv: 2 pooling: 2 FC: 1		
	LeNet[21]	99,3 %			6 conv: 3 Pooling: 2 FC: 1		

Table C.1 *(continued)*

Dataset	Model	Accuracy	Top-1 error	Top-5 error	Model depth	Parameters	Model parameters and values
CIFAR-10	Efficient-Net-B7[18]	98,9 %				64M	momentum: 0,9 weight decay: 0,0001 learning rate: 0,256
	ColorNet[22]	98,46 %				19,0M	momentum: 0,9 decay rate: 0,95 learning rate: 0,0001-0,001
	DenseNet[23]	96,54 %					batch size: 6 momentum: 0,9 weight decay: 0,0001 learning rate: 0,1 dropout rate: 0,2
LFW	FaceNet[24]	99,63 %				7,5M	learning rate: 0,05
	DeepID3[25]	99,53 %					
	DeepFace[26]	97,5 %					batch size: 128 learning rate: 0,01

Annex D
(informative)

Chance-corrected cause-specific mortality fraction

D.1 Calculating chance-corrected cause-specific mortality fraction accuracy

The cause-specific mortality fraction (CSMF) is the fraction of in-hospital deaths for a given cause normalized over all causes, where the underlying cause of death has been coded according to the International Classification of Diseases.[27] C_{SMF} accuracy is a measure of predictive quality at the population level, which quantifies how closely the estimated C_{SMF} values approximate the truth.

Given a confusion matrix M where $M_{\mathrm{I,J}}$ is the number of data samples with true class i and inferred (or predicted) class j,

$$C_{SMF}{}_i^t = \frac{\sum_{k=1}^{K} M_{\mathrm{i,k}}}{N}$$

and

$$C_{SMF}{}_j^p = \frac{\sum_{k=1}^{K} M_{\mathrm{k,j}}}{N}$$

where N is the total number of data samples,

$$N = \sum_{i}^{K} \sum_{j}^{K} M_{i,j}$$

then non-chance-corrected CSMF accuracy (C_{SMFA})with a minimum value (m) of j is:

$$C_{SMFA} = 1 - \frac{\left| C_{SMF}{}_i^t - C_{SMF}{}_j^p \right|}{2 * \left(1 - m_j \, C_{SMF}{}_i^t\right)}$$

When it is possible to compute its associated coefficients, the chance-corrected version of CSMF accuracy can be used[28].

Bibliography

[1] ISO/IEC TR 24027, *Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making*

[2] HAN L. Aaron. *LEPOR: An Augmented Machine Translation Evaluation Metric.* University of Macau, 2014 [viewed 21 September 2021]. Available from: https://library2.um.edu.mo/etheses/b33358400_ft.pdf

[3] HERNANDEZ D., BROWN T.P. *Measuring the Algorithmic Efficiency of Neural Networks*. 2020 [viewed 21 September 2021]. Available from: https://arxiv.org/pdf/2005.04305.pdf

[4] HAILO, A Practical Guide to Edge AI Power Efficiency. 2021, Israel. https://www.kisacoresearch.com/sites/default/files/documents/halio_whitepaper_edge_ai_power_efficiency.pdf

[5] JOHNSSON B., GANESTAM P., DOGGETT M., AKENINE-MÖLLER T. Power efficiency for software algorithms running on graphics processors. *Fourth ACM SIGGRAPH / Eurographics conference on High-Performance Graphics*. 2012, pages 67-75. https://diglib.eg.org/bitstream/handle/10.2312/EGGH.HPG12.067-075/067-075.pdf

[6] SZE V., CHEN Y.-H., YANG T.-J., EMER J.S. How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful. *IEEE Solid-State Circuits Magazine.* 2020, **12 (3)**, pages 28-41. https://ieeexplore.ieee.org/document/9177369

[7] AKENINE-MÖLLER Tomas, JOHNSSON Björn, Performance per what? Journal of Computer Graphics Techniques. 2012, **1**, pages 37-41. https://jcgt.org/published/0001/01/03/paper.pdf

[8] BLOUW P., CHOO X., HUNSBERGER E., ELIASMITH C. Benchmarking keyword spotting efficiency on neuromorphic hardware. *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. 2019, pages 1-8. https://arxiv.org/pdf/1812.01739.pdf

[9] STRUBELL E., GANESH A., MCCALLUM A. Energy and policy considerations for deep learning in NLP. *57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019. https://arxiv.org/pdf/1906.02243

[10] PATEL Yashwant Singh, JAIN Kalash, SHUKLA Surendra Kumar, A Brief Survey on Benchmarks and Research Challenges for Green Cloud Computing. International Journal of Computer Applications. 2016, **3**, pages 5-10. https://research.ijcaonline.org/ncacit2016/number3/ncacit3046.pdf

[11] STRUBELL E., GANESH A., MCCALLUM A., Energy and Policy Considerations for Deep Learning in NLP. The Thirty-Fourth AAAI Conference on Artificial Intelligence. 2020, **20**, pages 13693-13696. https://ojs.aaai.org/index.php/AAAI/article/view/7123/6977.

[12] HENDERSON Peter, HU Jieru, ROMOF Joshua, BRUNSKILL Emma, JURAFSKY Dan, PINEAU Joelle, Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. Journal of Machine Learning Research. 2020, **21**. https://www.jmlr.org/papers/volume21/20-312/20-312.pdf.

[13] DIETTERICH Thomas G. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation, Volume 10, Issue 7*. 1998, **10 (7)**, pages 1895-1923. https://doi.org/10.1162/089976698300017197

[14] WILCOXON Frank, Individual Comparisons by Ranking Methods. Biometrics Bulletin. 1945, **1 (6)**, pages 80-83.

[15] KRIZHEVSKY A., SUTSKEVER I., HINTON G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. 2012, **25**, pages 1097-1105.

https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[16] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations.* 2015, **3.** https://arxiv.org/pdf/1409.1556.pdf

[17] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition. 2016, pages 770-778. https://arxiv.org/pdf/1512.03385

[18] Tan Mingxing, Le Quoc, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning. 2019, pages 6105-6114. https://arxiv.org/pdf/1905.11946

[19] Kowsari K., Heidarysafa M., Brown D.E., Meimandi K.J., Barnes L.E. RMDL: Random Multimodel Deep Learning for Classification. *Proceedings of the 2nd International Conference on Information System and Data Mining*. 2018, **2**, pages 19-28. https://arxiv.org/pdf/1805.01890

[20] Ciregan Dan, Meier Ueli, Schmidhuber Juergen, Multi-Column Deep Neural Networks for Image Classification. IEEE Conference on Computer Vision and Pattern Recognition. 2012, pages 3642-3649. https://arxiv.org/pdf/1202.2745

[21] Lécun Yann, Bottou Leon, Bengio Yoshua, Haffner Patrick, Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 1998, **86(**11**)**, pages 2278-2324. http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[22] Zerman, Emin and, Rana, Aakanksha and Smolic, Aljosa. ColornNet – Estimating Colorfulness in Natural Images. *IEEE International Conference on Image Processing*. 2019, pages 3791-3795. https://arxiv.org/pdf/1908.08505

[23] Huang G., Zhuang L., van der Maaten L., Weinberger K.Q. Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pages 4700-4708. https://arxiv.org/pdf/1608.06993

[24] Schroff F., Kalenichenko D., Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pages 815-823. https://arxiv.org/pdf/1503.03832

[25] Sun Y., Liang D., Wang X., Tang X. *DeepID3: Face Recognition with Very Deep Neural Networks*. The Chinese University of Hong Kong, 2015 [viewed 21 September 2021]. https://arxiv.org/pdf/1502.00873

[26] Taigman Yaniv, Yang Ming and Ranzato, Marc'Aurelio and Wolf, Lior. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pages 1701-1708.

[27] Murray Christopher, Lopez Alan, Barofsky Jeremy, Bryson-Cahn Chloe, Lozano Rafael, Estimating Population Cause-Specific Mortality Fractions from in-Hospital Mortality: Validation of a New Method. PLoS Medicine. 2007, **4** (326**)**. https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.0040326

[28] Flaxman Abraham, Serina Peter, Bernardo Hernandez, Murray Christopher, Riley Ian, Lopez Alan, Measuring causes of death in populations: a new metric that corrects cause-specific mortality fractions for chance. Population Health Metrics. 2015, **13** (28**)**. https://pophealthmetrics.biomedcentral.com/track/pdf/10.1186/s12963-015-0061-1.pdf

[29] ISO 7200:2004, *Technical product documentation — Data fields in title blocks and document headers*

This Indian Standard has been developed from Doc No.: LITD 30 (22593).

Amendments Issued Since Publication

Amend No.	Date of Issue	Text Affected

Published by BIS, New Delhi