Metrics for Multi-class and Multi-label Classification

Maximilian Wenzel
maximilian.wenzel@uni-ulm.de
Institute of Databases and Information Systems, University Ulm
Germany

ABSTRACT

Classification metrics are used to summarize the performance of classifiers under different aspects which are relevant to the appropriate application domain. Multi-class and multi-label classification metrics are grounded on the binary classification case in that the metrics are initially computed per class or per instance for a specific data set and subsequently summarized by deploying a respective averaging strategy. However, there exists a multitude of averaging strategies which produce diverging results and may even lead to a different ranking of the evaluated classifiers regarding their performance. One particular example for this phenomenon is the macro-averaged F_1-score. We examine these metrics and averaging strategies in detail, explain why a strategy is applied in which case, and show their potential variation by accompanying examples. Proceeding from three recent machine learning papers, we highlight the best practice in dealing with evaluation metrics: The equations to compute the metrics should always be explicitly included and, if several classifiers are involved, the metrics should be calculated by a single authority as it is commonly the case in machine learning challenges to ensure a correct and consistent evaluation.

1 INTRODUCTION

Classification describes the problem of categorizing a given instance into a single class or multiple classes. In order to generate an appropriate classifier for such a problem, the classifier commonly undergoes a training and a testing phase. In the training phase, the classifier learns to predict the correct outcomes for given instances and, subsequently, in the testing phase, the actual performance of the classifier is evaluated by counting the number of correct and incorrect predictions. In order to summarize these absolute counts, evaluation metrics are deployed which can then be used to compare the performance of different classifiers to one another. However, since the concrete computation of specific metrics is not standardized and small variations of a metric may even result in a different classifier ranking, the correct and concise evaluation of classifiers should not be neglected and represents one of the central pillars in machine learning and data science.

In this survey, we examine the fundamental classification metrics from the multi-class and multi-label classification case. However, since both cases build upon binary classification, we introduce the essential metrics first for the binary case. We explicitly focus on the basic metrics which can be derived from the confusion matrix. Other metrics which probably are rather seldom used in practice or specific to a given application domain usually then can be easily derived.

We begin in Section 2 by introducing the formal definition of binary, multi-class, and multi-label classification. Section 3 subsequently introduces the appropriate metrics for the binary, multi-class, and multi-label case accompanied by concrete examples which shall illustrate their utilization in practice and their differences. After we looked at all metrics from the appropriate classification cases, Section 4 undertakes an excursion into practice, where three recent machine learning papers and their handling with evaluation metrics are presented. In Section 5, we discuss our investigation and Section 6 eventually concludes the paper.

2 PRELIMINARIES TO CLASSIFICATION

In the following section, a formal introduction to binary, multi-class, and multi label classification is provided in order to ensure a clear and consistent terminology. We start off with the case of binary classification as it lays the foundation for the multi-class and multi-label case. The formal notation has been primarily adopted from a combination of Tharwat [10], Sorower [9] and a proposal for a standardized notation in machine learning of the Beijing academy of artificial intelligence [3].

2.1 Binary Classification

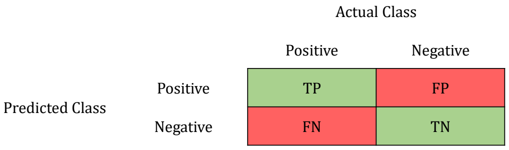
Definition 2.1 (Dataset and Hypothesis Function). Let $S = \{z_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$ be a dataset which represents a distribution $\mathcal{D}$ over a domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the instances domain, $\mathcal{Y}$ is the label domain, and $\mathcal{Z}$ is the example domain. The instance domain $\mathcal{X}$ is a subset of $\mathbb{R}^d$ and the label domain $\mathcal{Y}$ in turn is a subset of $\mathbb{R}^{d_o}$, where d, d_o is the input dimension and output dimension, respectively. The hypothesis function or classifier is depicted by $h \in \mathcal{H}$ with $h : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{H}$ represents the hypothesis space. Thus, given an instance $x_i \in \mathcal{X}$, $h(x_i) \in \mathcal{Y}$ corresponds to the predicted label of our classifier h which may not necessarily correspond to the actual label y_i.

Definition 2.2 (Binary Classification). In a binary classification scenario, the label domain is defined by $\mathcal{Y} = \{0, 1\}$, where each instance $x_i \in \mathcal{X}$ has assigned a label $y_i \in \mathcal{Y} = \{0, 1\}$. Therefore, we have $h : \mathcal{X} \rightarrow \{0, 1\}$. The appropriate labels $\mathcal{Y} = \{0, 1\}$ can be mapped to two distinct classes C_1, C_2.

The classifier function h for a given dataset $S = \{z_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$ thus induces altogether four distinct counts:

(1) *True positives* (TP): the number of instances x_i, where $h(x_i) = y_i = 1$
(2) *True negatives* (TN): the number of instances x_i, where $h(x_i) = y_i = 0$
(3) *False positives* (FP) / *Type I Error*: the number of instances x_i, where $h(x_i) = 1 \neq y_i = 0$
(4) *False negatives* (TP) / *Type II Error*: the number of instances x_i, where $h(x_i) = 0 \neq y_i = 1$

Subsequently, these measures can be summarized in terms of a 2×2 *confusion matrix/contingency table* which is depicted in Figure 1.



		Actual Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

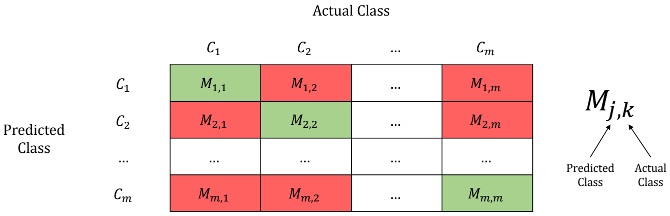
Figure 1: Confusion matrix for binary classification

2.2 Multi-class Classification

Definition 2.3 (Multi-class Classification). Let $C = \{C_1, C_2, ..., C_m\}$ be a set of classes, where $m = d_o$ corresponds to the output dimension of our label space $\mathcal{Y}$. In a multi-class classification scenario, the classifier h assigns to a given instance $x_i \in \mathcal{X}$ exactly one class $C_j \in C$, where $j \in \{1, \ldots, m\}$. Therefore, $\mathcal{Y} = \{\, y \,|\, y \in \{0,1\}^m, \sum_{k=1}^{m} y[k] = 1 \,\}$, i.e., exactly one entry j of the vector y is set to 1 (one-hot encoding) which represents the label for class C_j. We refer to $h(x_i)$ as the predicted class whereas y_i denotes the actual class.

In multi-class classification [5, 8, 10], we can map (analogously to the binary case) the predicted class of our classifier $h(x_i)$ to the actual class y_i and thus generate a $m \times m$ matrix M which is depicted in Figure 2. An entry $M_{j,k} \in \mathbb{N}_0$ counts the number of times the classifier h predicted the class C_j whereas the actual class corresponds to C_k with $j, k \in \{1, \ldots, m\}$. The diagonal $M_{j,j}$ subsequently counts the number of correctly classified instances whereas all other entries correspond to the cases of a wrong prediction. Therefore, for each class C_j an appropriate 2×2 confusion matrix can be derived in the following way which is additionally shown in Figure 3:

(1) *True positives* $TP = M_{j,j}$
(2) *True negatives* $TN = \sum_{i=1,i\neq j}^{m} \sum_{k=1,k\neq j}^{m} M_{i,k}$
(3) *False positives* $FP = \sum_{i=1,i\neq j}^{m} M_{j,i}$
(4) *False negatives* $FN = \sum_{i=1,i\neq j}^{m} M_{i,j}$



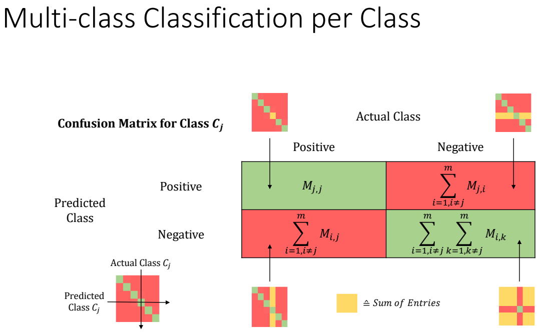
		Actual Class			
		C_1	C_2	...	C_m
Predicted Class	C_1	$M_{1,1}$	$M_{1,2}$	...	$M_{1,m}$
	C_2	$M_{2,1}$	$M_{2,2}$	...	$M_{2,m}$
	...	...	...	...	...
	C_m	$M_{m,1}$	$M_{m,2}$	...	$M_{m,m}$

$M_{j,k}$
Predicted Class
Actual Class

Figure 2: Confusion matrix for multi-class classification

2.3 Multi-Label Classification

Definition 2.4 (Multi-label Classification). In a multi-label classification scenario [2, 5, 8, 9, 13], the classifier h assigns to a given instance $x_i \in \mathcal{X}$ a set of classes $C_{h(x_i)} \subseteq C$. Thus, $\mathcal{Y} = \{\, y \,|\, y \in \{0,1\}^m \}$, i.e., multiple entries can be set to 1 and, analogously to the multi-class case, the entry j in the vector y represents the appropriate label for class C_j. We call $h(x_i)$ the predicted label set whereas y_i in turn denotes the actual label set, i.e., the predicted classes $C_{h(x_i)}$ and actual classes C_{y_i}, respectively.



Multi-class Classification per Class
Confusion Matrix for Class C_j
Actual Class
Positive
Negative
Predicted Class
Positive
Negative
$M_{j,j}$
$\sum_{i=1,i\neq j}^{m} M_{j,i}$
$\sum_{i=1,i\neq j}^{m} M_{i,j}$
$\sum_{i=1,i\neq j}^{m} \sum_{k=1,k\neq j}^{m} M_{i,k}$
Actual Class C_j
Predicted Class C_j
≙ Sum of Entries

Figure 3: Per class confusion matrix which can be derived from the original $m \times m$ multi-class confusion matrix

As in the multi-class case, we can generate a per-class confusion matrix which is also referred to as the *label-based* evaluation [9]. In this case the appropriate values for the confusion matrix for a given class C_j are derived in the following manner:

(1) *True positives* $TP = \sum_{i=1}^{n} |\{x_i \,|\, C_j \in C_{h(x_i)} \wedge C_j \in C_{y_i}\}|$
(2) *True negatives* $TN = \sum_{i=1}^{n} |\{x_i \,|\, C_j \notin C_{h(x_i)} \wedge C_j \notin C_{y_i}\}|$
(3) *False positives* $FP = \sum_{i=1}^{n} |\{x_i \,|\, C_j \in C_{h(x_i)} \wedge C_j \notin C_{y_i}\}|$
(4) *False negatives* $FN = \sum_{i=1}^{n} |\{x_i \,|\, C_j \notin C_{h(x_i)} \wedge C_j \in C_{y_i}\}|$

In contrast to the binary and multi-class case, the prediction of our classifier h can additionally be evaluated *per instance*. The classification of a given instance x_i thus can be *fully correct*, *partially correct*, and *fully incorrect* as presented below:

- *Fully correct*: $C_{h(x_i)} = C_{y_i}$
- *Partially correct*: $\exists c \in C_{h(x_i)} : c \in C_{y_i}$
- *Fully incorrect*: $\nexists c \in C_{h(x_i)} : c \in C_{y_i}$

Therefore, instead of generating a 2×2 confusion matrix per class, it can be also generated per instance x_i which is described in the following and additionally depicted in Figure 4:

(1) *True positives* $TP = |C_{h(x_i)} \cap C_{y_i}|$
(2) *True negatives* $TN = |(\mathcal{Y} \setminus C_{h(x_i)}) \cap (\mathcal{Y} \setminus C_{y_i})|$
(3) *False positives* $FP = |C_{h(x_i)} \setminus C_{y_i}|$
(4) *False negatives* $FN = |C_{y_i} \setminus C_{h(x_i)}|$

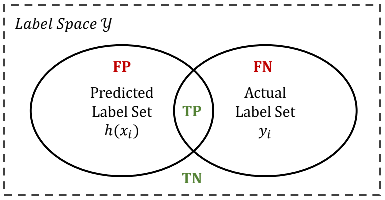


Figure 4: Multi-label classification confusion matrix which can be generated per instance x_i

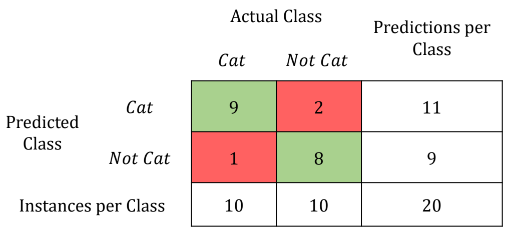
3 CLASSIFICATION METRICS

The performance of a classifier on a given dataset S can be assessed by considering the appropriate results from the confusion matrix, i.e., the TP, TN, FP, and FN values. The relation of these particular values to one another is subsequently summarized using *classification metrics*. In this section, we introduce and discuss the fundamental metrics of the multi-class and multi-label classification case. Since both cases are derived from the binary case, we start off with the binary classification metrics. In addition to it, the appropriate metrics are analyzed in regards to their sensitivity for *imbalanced data* [10].

Definition 3.1 (Imbalanced Data). Imbalanced data describes a dataset $S = \{z_i\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$ where the class distribution varies across different classes. The class distribution characterizes the ratio between the number of instances x_i which have a particular class $C_j \in \{C_1, \ldots, C_m\}$ assigned in relation to the overall number of instances n of a given dataset.

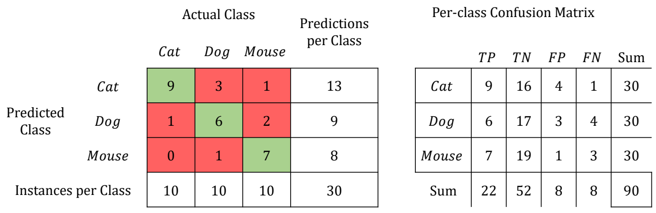
In principle, all metrics which consider values from the actual positive and the actual negative column of a given confusion matrix are sensitive to imbalanced data. Therefore, if the class distribution changes even though the classifier performance stays the same, i.e., the same proportion of instances per class has been classified correctly (or incorrectly), the result of the appropriate metric also changes which is shown in the upcoming examples.

Note that the issue of imbalanced data is naturally present in the case of a per-class confusion matrix of a given multi-class scenario. For instance, in Figure 6 an exemplary multi-class confusion matrix is depicted which may have its origin from an image classification task. As can be seen, the classes are equally distributed in that there is an equal amount of instances present for each class. However, if the per-class confusion matrix is generated, as shown in Figure 7, the derived classes, i.e., Cat and $NotCat$, are in turn not equally distributed. In contrast, Figure 5 shows an example of a purely binary classification matrix (independent from the previous example) with an equal class distribution but the same ratio between the TP and FN values and the FP and TN values. We will see how those two examples behave differently in correspondence to the deployed metric.



		Actual Class		Predictions per Class
		Cat	*Not Cat*	
Predicted Class	*Cat*	9	2	11
	Not Cat	1	8	9
Instances per Class		10	10	20

Figure 5: Example of a confusion matrix in a binary classification scenario with equal class distribution

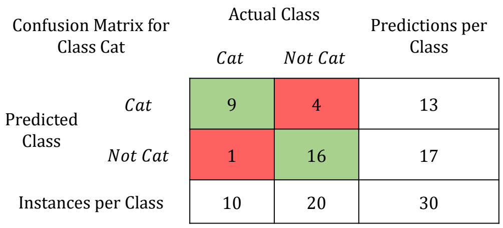


		Actual Class			Predictions per Class
		Cat	*Dog*	*Mouse*	
Predicted Class	*Cat*	9	3	1	13
	Dog	1	6	2	9
	Mouse	0	1	7	8
Instances per Class		10	10	10	30

Per-class Confusion Matrix

	TP	*TN*	*FP*	*FN*	Sum
Cat	9	16	4	1	30
Dog	6	17	3	4	30
Mouse	7	19	1	3	30
Sum	22	52	8	8	90

Figure 6: Example of a confusion matrix for a multi-class classification scenario with equal class distribution



Confusion Matrix for Class Cat		Actual Class		Predictions per Class
		Cat	*Not Cat*	
Predicted Class	*Cat*	9	4	13
	Not Cat	1	16	17
Instances per Class		10	20	30

Figure 7: Example of a per-class confusion matrix from the multi-class classification case of Figure 6

3.1 Binary Metrics

3.1.1 Sensitivity and Specificity. Sensitivity / recall / true positive rate (TPR) / *hit rate* refers to the proportion of instances which have been classified as positive in relation to the actual number of positive instances. In the binary classification case, the *recall r* therefore is computed in the following manner:

$$r = TPR = \frac{TP}{TP + FN} \tag{1}$$

Analogously, the *specificity* or *true negative rate* (TNR) calculates the ratio between the number of negative classified instances and the actual negative instances:

$$TNR = \frac{TN}{TN + FP} \tag{2}$$

Since both metrics only use values from one column, i.e., the column of the actual positive or the actual negative instances, the TPR and the TNR are insensitive to imbalanced data. For instance, in relation to the balanced dataset from Figure 5, we have a recall $r = \frac{9}{9+1} = \frac{9}{10}$, which corresponds to the result that we also get for the imbalanced dataset from Figure 7.

3.1.2 False Positive and False Negative Rates. The *false positive rate* (FPR) / *false alarm rate* / *fallout* and the *false negative rate* (FNR) / *miss rate* essentially represent the complement of the TPR and TNR, respectively. Therefore, these metrics are defined as follows:

$$FPR = 1 - TNR = \frac{FP}{FP + TN} \tag{3}$$

$$FNR = 1 - TPR = \frac{FN}{FN + TP} \tag{4}$$

Again, both metrics are computed using values from only one column of our confusion matrix and, hence, the FPR and the FNR are not sensitive to imbalanced data.

3.1.3 Predictive Values. The predictive values attempt to summarize the performance of a given classifier concerning either the positive or negative prediction. The *positive predictive value* (PPV) / *precision* computes how many of the positive classified instances are actually positive. The inverse of the PPV is given by the *false discovery rate* (FDR), which thus leads us to the following definition:

$$p = PPV = \frac{TP}{TP + FP} = 1 - FDR \tag{5}$$

Analogously, the *negative predictive value* (NPV) / *inverse precision* / *true negative accuracy* (TNA) represents the ratio between the negative instances which were classified correctly and the total number of negative predicted instances. In turn, the *false omission rate* (FOR) denotes the complement of the NPV.

$$NPV = \frac{TN}{TN + FN} = 1 - FOR \tag{6}$$

The predictive values consider values from both columns of the confusion matrix, i.e., from the actual positive and actual negative column and thus are sensitive to imbalanced data. This can be observed if we compute the precision p for our examples from Figure 5 and 7: In case of the balanced dataset, we have $p = \frac{9}{9+2} \approx 0.82$. However, in relation to imbalanced dataset, we get $p = \frac{9}{9+4} \approx 0.69$.

3.1.4 Accuracy and Error Rate. The *accuracy* summarizes the performance of a given classifier concerning the correctly classified instances in total, i.e., it combines the PPV and NPV. Hence, the metric is calculated in the following way:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

The inverse of the *accuracy* is defined by the *error rate* (ERR) / *misclassification rate* / *Hamming-Loss* and thus can be computed as follows:

$$ERR = 1 - Acc = \frac{FP + FN}{TP + TN + FP + FN} \tag{8}$$

The *accuracy* and *error rate* is sensitive to imbalanced data which can be seen especially in the multi-class and multi-label case which we consider in the following sections.

3.1.5 F-score. The *F-score* / *F-measure* summarizes the precision p and the recall r metric by computing a weighted variant of the harmonic mean, namely the *F_β-score*, which is given as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{p \cdot r}{(\beta^2 \cdot p) + r} \tag{9}$$

Most often $\beta = 1$ which is either denoted by the F_1-score or implicitly assumed when referring to the F-score. The F_1-score corresponds to the harmonic mean and thus the smaller value between p and r is weighted higher as can be seen in the following equation:

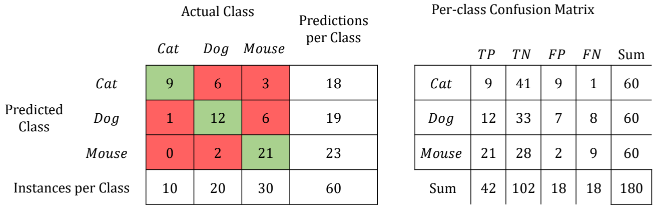
$$F_1 = \frac{2 \cdot p \cdot r}{p + r} = \left(\frac{p + r}{2 \cdot p \cdot r}\right)^{-1} = \left(\frac{p^{-1} + r^{-1}}{2}\right)^{-1} \tag{10}$$

3.2 Multi-class Metrics

Proceeding from the binary case, we now consider the introduced metrics in the context of a multi-class classification scenario [5, 8, 10]. In general, the aforementioned metrics are at first computed per class and, subsequently, averaged over all appropriate results. Basically, there are three distinct approaches for averaging the results of a given metric:

- **Macro-averaging**: The given metric is initially computed per class and, afterwards, the arithmetic mean is calculated from these acquired values.
- **Micro-averaging**: The absolute values in the numerator and denominator of the appropriate metric are summed up for all classes separately and the resulting fraction thus yields the outcome.
- **Weighted-averaging**: This approach represents a weighted variant of the macro-averaging strategy in that the appropriate metrics are computed initially per class and, afterwards, weighted according to their number of instances in the dataset.

Since in the case of micro-averaging, the absolute values are used, which might especially differ from class to class on imbalanced data, the results from classes with a higher number of instances are subsequently weighted higher [8]. The main difference between the micro-averaged and weighted-averaged strategy is that the weighted-averaged approach always weights the per-class result of the appropriate metric by the number of instances per class. This is however not always the case for the micro-averaged strategy since, e.g., in case of the precision metric, not the total number of instances per class are considered in the computation, but rather the total number of predictions per class as can be seen in Equation 5. In case of the macro-averaging approach, the arithmetic mean over all acquired per-class results is considered and, thus, all classes are treated equally in relation to the number of instances. On balanced datasets, the macro-averaging approach always yields the same result as the weighted-averaging strategy.



		Actual Class			Predictions per Class
		Cat	*Dog*	*Mouse*	
Predicted Class	*Cat*	🟩 9	🟥 6	🟥 3	18
	Dog	🟥 1	🟩 12	🟥 6	19
	Mouse	🟥 0	🟥 2	🟩 21	23
Instances per Class		10	20	30	60

Per-class Confusion Matrix

	TP	*TN*	*FP*	*FN*	Sum
Cat	9	41	9	1	60
Dog	12	33	7	8	60
Mouse	21	28	2	9	60
Sum	42	102	18	18	180

Figure 8: Example of a confusion matrix for a multi-class classification scenario with imbalanced class distribution

3.2.1 Recall, TNR, FPR, and FNR. By applying the metrics from the binary classification case for each class, we can hence compute the summarized metric over all classes by micro- or macro-averaging over the particular results. Since the recall, FPR, and FNR are insensitive to imbalanced data, they behave similarly and, hence, in the following, we consider only the recall metric.

Let r_μ, r_M, and r_w be the micro-, macro-, and weighted-averaged recall value, respectively, over a given set of classes $C = \{C_1, \ldots, C_m\}$. Furthermore, let TP_j, TN_j, FP_j, and FN_j be the appropriate values from the derived per-class confusion matrix for a given class $C_j \in C$. The micro-, macro-, and weighted-averaged recall is then defined as follows:

$$r_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m}(TP_j + FN_j)} \tag{11}$$

$$r_M = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{(TP_j + FN_j)} \tag{12}$$

$$r_w = \frac{1}{n}\sum_{j=1}^{m}(TP_j + FN_j)\cdot\frac{TP_j}{(TP_j + FN_j)} = \frac{1}{n}\sum_{j=1}^{m} TP_j \tag{13}$$

In relation to the example from Figure 6 with a balanced class distribution, we have $r_\mu = \frac{9+6+7}{9+1+6+4+7+3} = r_w = r_M = \frac{1}{3}(\frac{9}{9+1} + \frac{6}{6+4} + \frac{7}{7+3}) = \frac{11}{15} \approx 0.733$. As can be seen, $r_\mu = r_M = r_w$ if the classes are equally distributed. However if we consider the example with an imbalanced class distribution from Figure 8, we have $r_\mu = \frac{9+12+21}{9+1+12+8+21+9} = \frac{7}{10} = r_w = \frac{9+12+21}{60} \neq r_M = \frac{1}{3}(\frac{9}{9+1} + \frac{21}{12+8} + \frac{21}{21+9})$. Consequently, although the per-class recall results are the same in the balanced and imbalanced dataset example, the appropriate micro- and weighted-averaged results differ from the macro-averaged result since classes with more instances are weighted higher. The macro-averaged recall is equal in both examples since it is independent from individual class distributions.

The equality of the weighted- and micro-averaged approach is a unique feature of metrics which divide by the total number of instances per class. Also, as already indicated, it must be noted that the weighted-averaged strategy always corresponds to the macro-averaged result if the classes are equally distributed. This can be observed in Equation 13 since the coefficient $(TP_j + FN_j)$, i.e., the number of instances per class, in the sum is equal for all classes and thus can be extracted from the sum. This consequently evaluates to $\frac{TP_j+FN_j}{n} = \frac{1}{m}$.

3.2.2 Predictive Values. As discussed in Section 3.1.3, the predictive values summarize as the name suggests the predictive performance of a given classifier and are sensitive to imbalanced data because these metrics use values from the actual positive and actual negative column of the confusion matrix. Again, since the precision, NPV, FDR, and FOR, behave similarly, we consider the multi-class case only for the precision p:

$$p_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m}(TP_j + FP_j)} \tag{14}$$

$$p_M = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{(TP_j + FP_j)} \tag{15}$$

$$p_w = \frac{1}{n}\sum_{j=1}^{m}(TP_j + FN_j)\cdot\frac{TP_j}{TP_j + FP_j} \tag{16}$$

In relation to the example from Figure 6, the computation of the micro- and macro-averaged precision evaluates to $p_\mu = \frac{9+6+7}{9+4+6+3+7+1} = \frac{11}{15} \approx 0.733 \neq p_M = \frac{1}{3}(\frac{9}{9+4} + \frac{6}{6+3} + \frac{7}{7+1}) \approx 0.745$. In case of the example with an imbalanced class distribution from Figure 8, however, we have $p_\mu = \frac{9+12+21}{9+9+12+7+21+2} = \frac{7}{10} \neq p_M = \frac{1}{3}(\frac{9}{9+9} + \frac{12}{12+7} + \frac{21}{21+2}) \approx 0.682$.

Again, as in the case of the recall metric, we get different micro-averaged results since the absolute number of predictions per class, i.e., $(TP_j + FP_j)$, varies. However, we get also different macro-averaged results because the precision is sensitive to imbalanced data. Thus, we have different per-class precision values and, hence, the arithmetic mean yields a different result.

As mentioned before, in case of a balanced dataset, the weighted-averaged approach corresponds to the macro-averaged result and, therefore, $p_M = p_w \approx 0.745$. However, in consideration of the imbalanced dataset, we have $p_w = \frac{1}{60}\cdot(10\cdot\frac{9}{18} + 20\cdot\frac{12}{19} + 30\cdot\frac{21}{23}) \approx 0.750$ which is considerably higher than the micro- and macro-averaged result. As we can already observe by these results, the evaluation of a given classifier may greatly depend on the chosen metric.

3.2.3 Accuracy and Error Rate. The multi-class computation of the accuracy and the error rate is exemplary introduced by the accuracy Acc since both metrics, the accuracy and error rate, are in turn computed similarly.

$$\begin{aligned} Acc_\mu &= \frac{\sum_{j=1}^{m} TP_j + TN_j}{\sum_{j=1}^{m}(TP_j + TN_j + FP_j + FN_j)} = \frac{\sum_{j=1}^{m} TP_j + TN_j}{m\cdot n} \\ &= \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j + TN_j}{(TP_j + TN_j + FP_j + FN_j)} = Acc_M \end{aligned} \tag{17}$$

As can be seen, since the denominator for each class always sums up to the total number of instances, the micro- and macro-averaged values are in case of the accuracy always equal. Analogously to the predictive values, the accuracy is sensitive to imbalanced data and we thus expect different results concerning the macro-averaged accuracy Acc_M of an balanced and imbalanced dataset, as can be seen in the upcoming example. In relation to Figure 6, we have $Acc_\mu = Acc_M = \frac{22+52}{90} = \frac{7}{10}$. In case of the imbalanced dataset from Figure 8, in turn, we have $Acc_\mu = Acc_M = \frac{42+102}{180} = \frac{8}{10}$.

Note that the accuracy is higher in case of the imbalanced dataset although both examples only vary in the absolute number of instances per class however not in the ratio between the TP_j and the FN_j values for a given class C_j. Therefore, a different variant of the accuracy metric is deployed in the multi-label case, where the number of classes $|C|$ is naturally very high and, thus, the TN_j value yields for each class C_j a comparatively high result in relation to the TP_j value.

The weighted averaging approach is usually not used in literature for the accuracy metric because the per-class TN value depends on the number of instances of all other classes and hence the accuracy weighted by the number of instances of a particular class is probably not reasonable.

3.2.4 F-score. In the following, we consider the multi-class case of the F-score. Since usually the F_1-score is deployed in practice, we introduce the metric for the case where $\beta = 1$. The calculation of the micro-averaged F_1-score is straightforward and can be accomplished in the following manner, where p_μ and r_μ represents the micro-averaged precision and recall, respectively:

$$F1_\mu = \frac{2\cdot p_\mu\cdot r_\mu}{p_\mu + r_\mu} \tag{18}$$

In comparison to our previously considered multi-class metrics, the approach to compute the macro-averaged F_1-score is not self evident since there exist two distinct strategies which we denote as $\mathcal{F}_1$ and $\mathbb{F}_1$ as it is introduced by Opitz and Burst [4]:

- $\mathcal{F}_1$, *the averaged* F_1: Let p_j, r_j be the precision and recall value of a given class $C_j \in C$. First, the F_1-score is computed per-class and, subsequently, the arithmetic mean is calculated from the per-class F_1-scores:

$$\mathcal{F}_1 = \frac{1}{m} \sum_{j=1}^{m} \frac{2 \cdot p_j \cdot r_j}{p_j + r_j} \tag{19}$$

- $\mathbb{F}_1$, *the* F_1 *of averages*: First, the macro-averaged precision p_M and recall r_M is computed and, afterwards, the appropriate results are deployed in the usual F_1 equation:

$$\mathbb{F}_1 = \frac{2 \cdot p_M \cdot r_M}{p_M + r_M} \tag{20}$$

In consideration of our example from Figure 6, we have

$$F_{1\mu} = \frac{2 \cdot \frac{7}{10} \cdot \frac{11}{15}}{\frac{7}{10} + \frac{11}{15}} = \frac{11}{15} \approx 0.733, \quad \mathbb{F}_1 = \frac{2 \cdot 0.745 \cdot \frac{11}{15}}{0.745 + \frac{11}{15}} \approx 0.739$$

$$\mathcal{F}_1 = \frac{1}{3}\left(\frac{2 \cdot \frac{9}{13} \cdot \frac{9}{10}}{\frac{9}{13} + \frac{9}{10}} + \frac{2 \cdot \frac{6}{9} \cdot \frac{6}{10}}{\frac{6}{9} + \frac{6}{10}} + \frac{2 \cdot \frac{7}{8} \cdot \frac{7}{10}}{\frac{7}{8} + \frac{7}{10}}\right) \approx 0.731$$

On the other hand, in case of our imbalanced example from Figure 8, we have:

$$F_{1\mu} = \frac{2 \cdot \frac{7}{10} \cdot \frac{7}{10}}{\frac{7}{10} + \frac{7}{10}} = \frac{7}{10}, \quad \mathbb{F}_1 = \frac{2 \cdot 0.682 \cdot \frac{11}{15}}{0.682 + \frac{11}{15}} \approx 0.706$$

$$\mathcal{F}_1 = \frac{1}{3}\left(\frac{2 \cdot \frac{9}{18} \cdot \frac{9}{10}}{\frac{9}{18} + \frac{9}{10}} + \frac{2 \cdot \frac{12}{19} \cdot \frac{12}{20}}{\frac{12}{19} + \frac{12}{20}} + \frac{2 \cdot \frac{21}{23} \cdot \frac{21}{30}}{\frac{21}{23} + \frac{21}{30}}\right) \approx 0.684$$

As it is the case for the other micro- and macro-averaged metrics, the micro-averaged $F_{1\mu}$ weights classes with more instances higher than smaller classes since the absolute values are deployed. Now, when it comes to $\mathcal{F}_1$ and $\mathbb{F}_1$ we can observe the following behavior: If the per-class precision p_j and recall r_j do not differ considerably between classes, the resulting difference between $\mathcal{F}_1$ and $\mathbb{F}_1$ is negligible. However, if this is not the case, the results may diverge in such a way that the evaluated classifiers get assigned different rankings depending on the deployed F-score.

Another aspect is the compensation of low, per-class p_j and r_j values in case of the $\mathbb{F}_1$-score: Individual values do not influence the overall results p_M and r_M as much as they would in case of the $\mathcal{F}_1$-score. Hence, the $\mathbb{F}_1$-score may be overly benevolent. Opitz and Burst [4] therefore recommend using the $\mathcal{F}_1$ metric, i.e., the arithmetic mean of the per-class F_1-scores, since it assigns an equal weight to all classes from a given dataset.

The SciKit-learn Python library [5] also provides a weighted-averaged F_1-score which uses the $\mathcal{F}_1$ approach and is thus defined as follows:

$$\mathcal{F}_{1w} = \frac{1}{n} \sum_{j=1}^{m} (TP_j + FN_j) \cdot \frac{2 \cdot p_j \cdot r_j}{p_j + r_j} \tag{21}$$

In case of the balanced dataset $\mathcal{F}_{1w}$ in turn is equal to $\mathcal{F}_1$. However, for the imbalanced dataset the equation evaluates to the following result:

$$\mathcal{F}_{1w} = \frac{1}{60} \left(10 \cdot 0.643 + 20 \cdot 0.615 + 30 \cdot 0.793\right) \approx 0.709$$

3.2.5 Relation between Micro-Precision, Micro-Recall, and Micro-F_1. In consideration of our previous examples, it stands out that the micro-precision, micro-recall, and micro-F_1 metric are equivalent in terms of their particular outcomes. A given entry $M_{i,j}$ in a multi-class confusion matrix M counts the number of times a classifier predicted the class C_i instead of the actual class C_j. From the perspective of class C_i it is a FP prediction, however, in relation to C_j it is a FN prediction. Thus the appropriate value $M_{i,j}$ is used both in the count of FP_i and FN_j and we therefore have

$$\sum_{j=1}^{m} FP_j = \sum_{j=1}^{m} FN_j \tag{22}$$

It thus follows that the micro-recall and micro-precision are equivalent. And since the micro-F_1-score uses the harmonic mean of the micro-recall and micro-precision metric, it correspondingly yields the same result.

3.3 Multi-label Metrics

We now consider analogously to the multi-class case the multi-label classification metrics [2, 5, 8, 9, 13] and, again, proceed from the initially introduced binary classification metrics. As described in Section 2.3, the confusion matrix can be generated per-class or per instance. Therefore, there are two distinct approaches to evaluate a multi-label classifier:

- *Per class*: Generate for each class an appropriate per-class confusion matrix and apply the macro, micro, or weighted averaging strategy over all per-class results of a given metric.
- *Per instance / sample*: Generate for each instance an appropriate confusion matrix and apply the macro-averaging strategy in order to summarize all results.

The question rises, why, in general, only the macro-averaging strategy should be used for the per-instance summary. Self-evidently, the weighted-averaged strategy cannot be applied because *weighted* always relates to the number of instances per class which cannot be included in this case. Furthermore, because we aim at weighting all instances equally, we cannot use the micro-averaging strategy. For instance, in case of the micro-averaged recall, the prediction of a particular instance would be weighted by the size of the actual label set C_{y_i}. Altogether, we therefore can only deploy the macro-averaging strategy in case of the per-instance approach.

Since the per-class summary essentially is the same for the multi-class and multi-label case after the confusion matrices have been generated as explained in Section 2.3, we will consider in the following only the per-instance approach with the appropriate macro-averaging strategy.

3.3.1 Recall, TNR, FPR, and FNR. Similar to the multi-class case, we can introduce the recall, TNR, FPR, and FNR together and, correspondingly, we use the *recall* metric again as representative for this group. The per-instance macro-averaged recall r_M over a dataset $S = \{z_i\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$ is thus defined as follows:

$$r_M = \frac{1}{n} \sum_{i=1}^{n} \frac{|C_{h(x_i)} \cap C_{y_i}|}{|C_{y_i}|} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i} \tag{23}$$

As depicted in Figure 9, we initially compute for each instance the appropriate confusion matrix and use these resulting values in our

Label Space $\mathcal{Y} = \{a, b, c, d, e, f, g\}$

Instance	*Predicted Label Set*	*Actual Label Set*	*TP*	*TN*	*FP*	*FN*	*Sum*
x_1	$\{a, b, c\}$	$\{a, b, c\}$	3	4	0	0	7
x_2	$\{a, b, d, e\}$	$\{a, b, c, d, e\}$	4	2	0	1	7
x_3	$\{e, f\}$	$\{c, d\}$	0	3	2	2	7
x_4	$\{b, c, d\}$	$\{a, c, d, g\}$	2	2	1	2	7
x_5	$\{a, c, d, f, g\}$	$\{g\}$	1	2	4	0	7
		Sum	10	13	7	5	35

Figure 9: Example of the appropriate TP, TN, FP, and FN values of a multi-label classifier over a given dataset

equation. Hence, we have $r_M = \frac{1}{5} \cdot (\frac{3}{3+0} + \frac{4}{4+1} + \frac{0}{0+2} + \frac{2}{2+2} + \frac{1}{1+0}) = 0.66$.

3.3.2 Predictive Values. In place of the predictive values, we once again introduce the precision metric for the per-instance summarized approach:

$$p_M = \frac{1}{n} \sum_{i=1}^{n} \frac{|C_{h(x_i)} \cap C_{y_i}|}{|C_{h(x_i)}|} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i} \tag{24}$$

In consideration of the example from Figure 9, we have $p_M = \frac{1}{5} \cdot (\frac{3}{3+0} + \frac{4}{4+0} + \frac{0}{0+2} + \frac{2}{2+1} + \frac{1}{1+4}) \approx 0.573$.

3.3.3 Exact Match Ratio. The *exact match ratio* (MR) / *subset accuracy* is a metric which is specific to multi-label classification. Given a dataset $S = \{z_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$, the MR counts the fully correct predicted label sets in relation to the total number of instances, i.e., all label sets $C_{h(x_i)}$ are counted, where $C_{h(x_i)} = C_{y_i}$ for a given instance $z_i \in S$. Naturally, this metric might be harsh especially in the case if the label space $\mathcal{Y}$ is large. The definition is as follows:

$$MR = \frac{1}{n} \sum_{i=1}^{n} I(C_{h(x_i)}, C_{y_i}), \quad \text{where} \quad I = \begin{cases} 1 & \text{if} \quad C_{h(x_i)} = C_{y_i} \\ 0 & otherwise \end{cases} \tag{25}$$

In consideration of our example from Figure 9, we thus get a MR of $\frac{1}{5}$ because only in case of x_1 the predicted label set is fully correct.

3.3.4 Accuracy. In the per-instance summarized multi-label case, the *Jaccard similarity* is computed per instance and, subsequently, the arithmetic mean is calculated over all appropriate results:

$$Acc = \frac{1}{n} \sum_{i=1}^{n} \frac{|C_{h(x_i)} \cap C_{y_i}|}{|C_{h(x_i)} \cup C_{y_i}|} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i + FN_i} \tag{26}$$

As can be observed in the equation, the Jaccard similarity leaves out the TN_i values in the nominator and denominator because, in a multi-label classification scenario, usually a high number of classes $|C|$ is deployed. If the usual macro-averaged accuracy from the multi-class case would be deployed, the TN_i value and, correspondingly, the overall accuracy would become comparatively high even if $C_{h(x_i)} = \emptyset$ since the actual label set C_{y_i} usually contains only a small proportion of labels from the label space $\mathcal{Y}$.

In relation to our example from Figure 9, we thus get $Acc = \frac{1}{5} \cdot (\frac{3}{3+0+0} + \frac{4}{4+0+1} + \frac{0}{0+2+2} + \frac{2}{2+1+2} + \frac{1}{1+4+0}) = 0.48$.

3.3.5 Hamming Loss. In the multi-class case, the *error rate* complements the accuracy and is defined by $ERR = 1 - Acc$. However, in multi-label classification, the *error rate* is called the *Hamming Loss* HL because it corresponds to the averaged *hamming distance* between the predicted label vectors and the actual label vectors. The *Hamming Loss* thus indicates how often on average a given class label was predicted incorrectly and is defined in the following manner:

$$\begin{aligned} HL =& \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|C|} |\{c \mid (c \in C_{h(x_i)} \land c \notin C_{y_i}) \lor (c \notin C_{h(x_i)} \land c \in C_{y_i})\}| \\ =& \frac{1}{n} \sum_{i=1}^{n} \frac{FP_i + FN_i}{TP_i + TN_i + FP_i + FN_i} \end{aligned} \tag{27}$$

By applying the given equation to our example from Figure 9, we thus have $\frac{1}{5} \cdot \frac{0+1+4+3+4}{7} = \frac{12}{35} \approx 0.343$.

3.3.6 F-score. If we generate the confusion matrix per class for a given multi-label classifier, we can deploy the four distinct approaches to calculate the F-score which were already discussed in Section 3.2.4 for the multi-class case. In the per-instance approach, we can proceed analogously: Either, we compute the $\mathcal{F}_1$ value, i.e., the average of the per-instance F_1-scores, or we compute the $\mathbb{F}_1$ value, i.e., the F_1-score of all per-instance macro-averaged precision and recall values. Thus, we have:

$$\mathcal{F}_1 = \frac{1}{n} \sum_{i=1}^{n} \frac{2 \cdot p_i \cdot r_i}{p_i + r_i} \tag{28}$$

$$\mathbb{F}_1 = \frac{2 \cdot p_M \cdot r_M}{p_M + r_M} \tag{29}$$

Judging by the prevalent literature for multi-label classification metrics [5, 8, 9, 13], the $\mathcal{F}_1$-score is the standard averaging approach. Concerning the example from Figure 9, we thus have $\mathcal{F}_1 = \frac{1}{5} \cdot (2 \cdot \frac{1 \cdot 1}{1+1}) + (2 \cdot \frac{1 \cdot 0.8}{1+0.8}) + 0 + (2 \cdot \frac{0.66 \cdot 0.5}{0.66+0.5}) + (2 \cdot \frac{0.2 \cdot 1}{0.2+1}) \approx 0.56$ and $\mathbb{F}_1 = (2 \cdot \frac{0.57 \cdot 0.66}{0.57+0.66}) \approx 0.61$. As can be seen, the difference between the $\mathcal{F}_1$-score and $\mathbb{F}_1$-score is about 0.05 because individual precision and recall values of specific instances are weighted higher. Certainly, this could also result in a higher $\mathcal{F}_1$-score than $\mathbb{F}_1$-score depending on the individual case.

4 RELATED WORK

In literature, the predominant multi-class [5, 8, 10] and multi-label [2, 5, 8, 9, 13] classification metrics are presented in appropriate surveys that summarize which metrics have been yet proposed in the field of data science. The actually deployed metrics may vary between each application domain, but, over the years, a best practice has evolved for the handling of metrics in order to avoid the associated pitfalls. In the following section, we want to highlight this best practice by the example of three recent publications in the machine learning field.

BioASQ[1] is a network which organizes challenges on biomedical semantic indexing and question answering in order to promote

[1] http://bioasq.org/

successful systems and methodologies in this particular domain. The *MESINESP task* [7] represents one of those challenges and deals with medical semantic indexing in Spanish. In the medical domain, querying of specialized information on a particular subject depends on complex semantic search capabilities, e.g., in order to retrieve relevant systematic reviews. Thus, experts from the medical domain increasingly rely on appropriate systems which provide sophisticated semantic search queries that deploy terms from a standardized vocabulary. In the *MESINESP task*, the challenge participants must assign labels to new medical documents from the DeCS vocabulary[2] in order to be queried afterwards by an appropriate information retrieval system. This annotation process has previously been done manually and was therefore labor-intensive and costly. The MESINESP task was then organized as follows: The participants of the challenge implemented their appropriate classifier and evaluated it over a given test dataset. The predictions for the test dataset were then delivered in JSON format to the appropriate committee which subsequently computed the per-class micro F_1-measure and the Lowest Common Ancestor F_1-measure (specific to hierarchically structured datasets). The metric was calculated centrally in order to prevent misconceptions and to ensure a consistent evaluation of the particular systems. In the actual presentation of the results, other evaluation metrics were also included which however were not considered in the eventual rating of the classifiers. These additional metrics include the macro-averaged F_1-score, however, it is not indicated (e.g., by an explicit equation) whether the $\mathcal{F}_1$ or $\mathbb{F}_1$ metric has been used. This presents us with the problem from Section 3.2.4. How important the coherent evaluation of all classifiers is shows a foot note in the MESINSEP task paper [7]: One participating team scored comparatively low on the determining micro F_1-measure but had, on the other hand, the highest score on the per-instance precision, macro precision, and micro precision.

Chen et al. [1] proposed a framework called *Semantic-Specific Graph Representation Learning (SS-GRL)* which can be used in order to enhance the performance of multi-label image classifiers by incorporating label dependency and semantic-aware regions. The framework essentially consists of two modules: A *semantic decoupling module* which extracts feature maps from images and a *semantic interaction module* that subsequently correlates the statistical label co-occurrence and explores the corresponding interactions using a graph propagation mechanism. All metrics which have been deployed in their evaluation are explicitly given by appropriate equations. They used two approaches in the evaluation of their classifiers: the first approach considers all labels of the prediction and the other approach evaluates only the labels with the top-3 highest prediction scores. These labels were subsequently compared with the ground truth labels and, afterwards, the per-instance micro precision, micro recall and micro F_1-score were computed. On the other hand, they also included the macro-averaged per-class evaluation, i.e., the recall, precision, and the F_1 of averages ($\mathbb{F}_1$-score). Utilizing a micro averaging strategy for a per-instance evaluation is however an issue as we discussed in Section 3.3 since the appropriate instances are weighted differently depending on the number of ground truth labels. If the number of ground truth labels is equal in every instance, the micro averaging strategy can be applied - this is the case for the evaluation which includes only those labels that have the top-3 highest score. In the case where all labels are included, this is however not the case. Chen et al. reference this issue but nevertheless include the micro-averaged results for the *all-labels* evaluation. As common for multi-label image detection tasks, the mean average precision (mAP) was additionally included which computes the intersection over union (IoU) of the predicted bounding box as well as the ground truth bounding box and averages over all classes. In their experiments on the PASCAL VOC 2007 & 2012, Microsoft COCO, and Visual Genome benchmarks, the SS-GRL framework outperformed current state-of-the-art methods by up to 6.7% in the mAP.

Traditional question answering (QA) systems over Linked Data and RDF either transform the given questions into triple patterns which are subsequently resolved using a triple store or use hybrid QA systems over RDF and text. All of these QA systems have their unique strengths and weaknesses. Usbeck et al. [12] therefore proposed a metasystem for question answering (QA) which integrates several QA systems: If a question is given to the QA metasystem, it subsequently decides which QA system is most qualified to answer the question at hand. The selection of a QA system corresponds to a multi-label classification problem and thus Usbeck et al. investigated which classifier might be best suited for this particular task by analyzing 16 classifiers from the MEKA framework [6] on the QALD-6 data set [11]. In order to compare these deployed classifiers to one another, they performed a 10-fold cross-validation and documented the macro $\mathcal{F}_1$-score on each test fold. The computation of the metric is again explicitly presented in their paper. Proceeding from the evaluation scores, they computed Cramér's V-coefficient which is based on Pearson's chi-squared statistic in order to determine the correlation of the classifier performance between specific topics, such as the topic *Location* or *Person*. The resulting metasystem subsequently outperformed the current state of the art in the QALD-6 benchmark.

5 DISCUSSION

Figure 10 summarizes our introduced metrics and the different averaging strategies. Of course, not all discussed metrics could be included in the depiction, however, we attempted to provide for each group of related metrics one representative measure which is also commonly deployed in practice.

The lack of standardization for the usage of evaluation metrics in specific tasks results in misconceptions regarding how a specific metric is computed and in a poor comparability of the resulting classifiers. However, it is not easy to centrally define a metric for specific tasks as can be seen for instance in the domain of sentiment analysis, subjectivity, and opinion, where there is no common consent on the deployed metrics [8]. A common consent for a standardized metric in specific tasks becomes even more difficult if the corresponding differences are more subtle, as for example in case of the $\mathcal{F}_1$- and $\mathbb{F}_1$-score. The $\mathcal{F}_1$-score computes the averages of the per-instance or per-class F_1-scores which weights individual precision and recall values higher. On the other hand, the $\mathbb{F}_1$ score uses the macro-averaged precision and recall values. Thus, although the intention of both metrics is almost the same, the eventual choice

[2] http://red.bvsalud.org/decs/en/about-decs/

can influence the evaluation of classifiers to such a degree, that they are ranked differently regarding their performance [4]. Therefore, as presented in Section 4, machine learning challenges are a solution to the problem in that the participants evaluate their classifiers themselves, however, they do not compute the resulting metrics. The metrics are usually centrally calculated by the appropriate committee and, hence, even if, e.g., it is not entirely clear which specific macro-averaged F_1-score has been deployed, the individual classifiers can still be compared to one another since a consistent evaluation approach has been used. Altogether, we can summarize the following points as best practice when dealing with metrics for the evaluation of classifiers:

- All deployed metrics in a classifier evaluation should always be explicitly given by an appropriate equation.
- If possible, the evaluation over the test dataset with all predicted and actual classes should be included in a structured format such as JSON.
- If an external library is utilized, e.g., the SciKit-learn Python library, it should be derived from the accompanying documentation or from the actual code how the metric is computed.
- In case of machine learning challenges, it is safest to collect the test dataset evaluations (instance ID, predicted classes, and actual classes) from all participants and subsequently compute the metrics centrally in order to provide a consistent comparison.

6 CONCLUSION

In this survey, we introduced and discussed the essential binary, multi-class, and multi-label classification metrics. We started off with a formal definition of the respective classification problems and introduced the confusion matrices which can be derived from the TP, TN, FP, and FN values of a classifier evaluation. These counts can subsequently be summarized by metrics which assess the overall performance of classifiers concerning different aspects. In the multi-class case, a binary confusion matrix is generated over a given set of instances for each class and, afterwards, the appropriate per-class results are summarized using the micro-, macro-, or weighted-averaging strategy. In multi-label classification, a classifier can not only be evaluated per class but also per instance. In the per-class approach, the proceeding is analogously defined to the multi-class case. However, in case of the per-instance approach, only the macro-averaging strategy can be used because all instances should be weighted equally. We presented three recent papers from the machine learning domain and examined their handling with evaluation metrics. As best practice, the evaluation over the test dataset with the predicted classes should be included and the utilized metrics should always be explicitly defined in equations. As can be concluded from our investigation, evaluation metrics are a central part of data science and, in order to avoid the associated pitfalls of multi-class and multi-label metrics, it is necessary to be attentive when evaluating or comparing classifiers.

REFERENCES

[1] Tianshui Chen, Muxin Xu, Xiaolu Hui, Hefeng Wu, and Liang Lin. 2019. Learning Semantic-Specific Graph Representation for Multi-label Image Recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 522–531.

[2] Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S Dhillon. 2015. Consistent Multilabel Classification. In *NIPS*, Vol. 29. 3321–3329.

[3] Beijing Academy of Artificial Intelligence. 2020. Suggested Notation for Machine Learning. https://github.com/Mayuyu/suggested-notation-for-machine-learning.

[4] Juri Opitz and Sebastian Burst. 2019. Macro F1 and Macro F1. *arXiv preprint arXiv:1911.03347* (2019).

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[6] Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoffrey Holmes. 2016. MEKA: A Multi-label/Multi-target Extension to WEKA. (2016).

[7] Carlos Rodriguez-Penagos, Anastasios Nentidis, Aitor Gonzalez-Agirre, Alejandro Asensio, Jordi Armengol-Estapé, Anastasia Krithara, Marta Villegas, Georgios Paliouras, and Martin Krallinger. 2020. Overview of MESINESP8, a Spanish Medical Semantic Indexing Task within BioASQ 2020. (2020).

[8] Marina Sokolova and Guy Lapalme. 2009. A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manag.* 45, 4 (2009), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

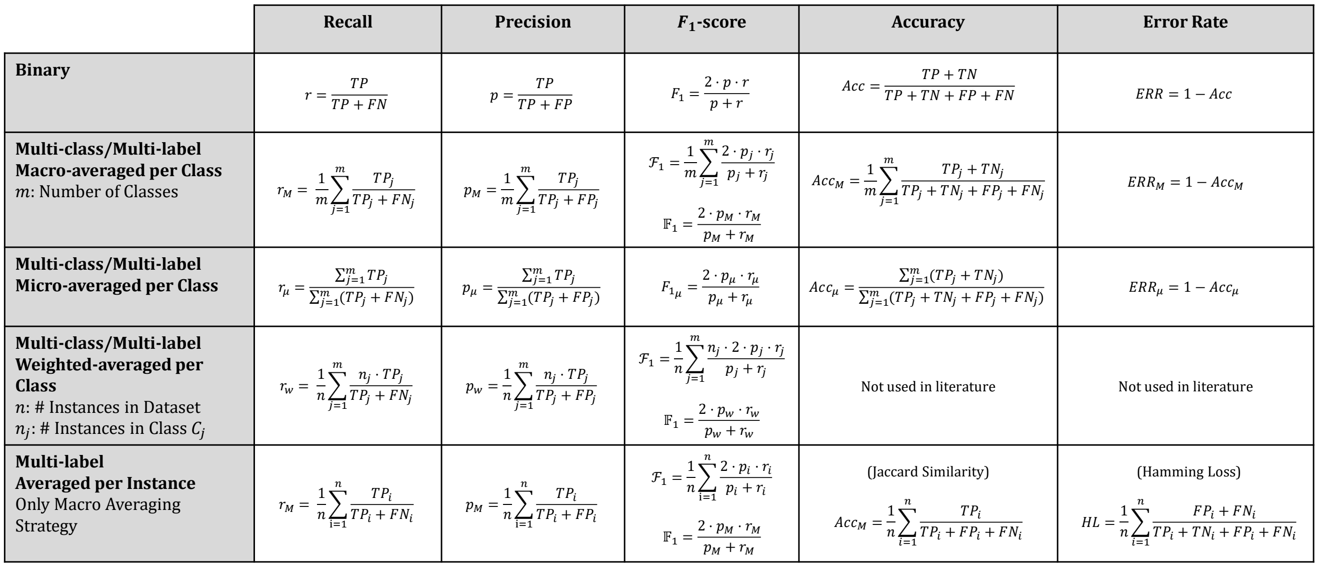
[9] Mohammad S Sorower. 2010. A Literature Survey on Algorithms for Multi-label Learning. *Oregon State University, Corvallis* 18 (2010), 1–25.

[10] Alaa Tharwat. 2020. Classification Assessment Methods. *Applied Computing and Informatics* (2020).

[11] Christina Unger, Axel-Cyrille Ngonga Ngomo, and Elena Cabrio. 2016. 6th Open Challenge on Question Answering over Linked Data (QALD-6). In *Semantic Web Evaluation Challenge*. Springer, 171–177.

[12] Ricardo Usbeck, Michael Hoffmann, Michael Röder, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. 2017. Using Multi-label Classification for Improved Question Answering. *arXiv preprint arXiv:1710.08634* (2017).

[13] Min-Ling Zhang and Zhi-Hua Zhou. 2013. A Review on Multi-label Learning Algorithms. *IEEE transactions on knowledge and data engineering* 26, 8 (2013), 1819–1837.



	Recall	**Precision**	**F_1-score**	**Accuracy**	**Error Rate**
Binary	$r = \frac{TP}{TP + FN}$	$p = \frac{TP}{TP + FP}$	$F_1 = \frac{2 \cdot p \cdot r}{p + r}$	$Acc = \frac{TP + TN}{TP + TN + FP + FN}$	$ERR = 1 - Acc$
Multi-class/Multi-label **Macro-averaged per Class** m: Number of Classes	$r_M = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{TP_j + FN_j}$	$p_M = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{TP_j + FP_j}$	$\mathcal{F}_1 = \frac{1}{m}\sum_{j=1}^{m}\frac{2 \cdot p_j \cdot r_j}{p_j + r_j}$ $\mathbb{F}_1 = \frac{2 \cdot p_M \cdot r_M}{p_M + r_M}$	$Acc_M = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j + TN_j}{TP_j + TN_j + FP_j + FN_j}$	$ERR_M = 1 - Acc_M$
Multi-class/Multi-label **Micro-averaged per Class**	$r_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m}(TP_j + FN_j)}$	$p_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m}(TP_j + FP_j)}$	$F_{1_\mu} = \frac{2 \cdot p_\mu \cdot r_\mu}{p_\mu + r_\mu}$	$Acc_\mu = \frac{\sum_{j=1}^{m}(TP_j + TN_j)}{\sum_{j=1}^{m}(TP_j + TN_j + FP_j + FN_j)}$	$ERR_\mu = 1 - Acc_\mu$
Multi-class/Multi-label **Weighted-averaged per Class** n: # Instances in Dataset n_j: # Instances in Class C_j	$r_w = \frac{1}{n}\sum_{j=1}^{m}\frac{n_j \cdot TP_j}{TP_j + FN_j}$	$p_w = \frac{1}{n}\sum_{j=1}^{m}\frac{n_j \cdot TP_j}{TP_j + FP_j}$	$\mathcal{F}_1 = \frac{1}{n}\sum_{j=1}^{m}\frac{n_j \cdot 2 \cdot p_j \cdot r_j}{p_j + r_j}$ $\mathbb{F}_1 = \frac{2 \cdot p_w \cdot r_w}{p_w + r_w}$	Not used in literature	Not used in literature
Multi-label **Averaged per Instance** Only Macro Averaging Strategy	$r_M = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FN_i}$	$p_M = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i}$	$\mathcal{F}_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2 \cdot p_i \cdot r_i}{p_i + r_i}$ $\mathbb{F}_1 = \frac{2 \cdot p_M \cdot r_M}{p_M + r_M}$	(Jaccard Similarity) $Acc_M = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i + FN_i}$	(Hamming Loss) $HL = \frac{1}{n}\sum_{i=1}^{n}\frac{FP_i + FN_i}{TP_i + TN_i + FP_i + FN_i}$

Figure 10: Summary of the fundamental binary, multi-class, and multi-label classification metrics and the distinct averaging strategies