Ann Math Artif Intell (2017) 81:429–450
DOI 10.1007/s10472-017-9564-8

A Bayesian interpretation of the confusion matrix

Olivier Caelen[1]

Published online: 11 September 2017

Abstract We propose a way to infer distributions of any performance indicator computed from the confusion matrix. This allows us to evaluate the variability of an indicator and to assess the importance of an observed difference between two performance indicators. We will assume that the values in a confusion matrix are observations coming from a multinomial distribution. Our method is based on a Bayesian approach in which the unknown parameters of the multinomial probability function themselves are assumed to be generated from a random vector. We will show that these unknown parameters follow a Dirichlet distribution. Thanks to the Bayesian approach, we also benefit from an elegant way of injecting prior knowledge into the distributions. Experiments are done on real and synthetic data sets and assess our method's ability to construct accurate distributions.

Keywords Confusion matrix · Classification · Bayesian statistics

Mathematics Subject Classfication (2010) 68T01 · 62G07 · 62F15

1 Introduction

The confusion matrix [1, 9] is typically used in machine learning to evaluate or to visualize the behavior of models in supervised classification contexts [7]. It is a square matrix in which the rows represent the actual class of the instances and the columns their predicted class. If we are handling a binary classification task, then the confusion matrix is a 2×2

✉ Olivier Caelen
olivier.caelen@worldline.com

[1] R & D, High Processing and Volume, Worldline S.A., Brussels, Belgium

matrix that reports the number of *true positives* ($\#TP$), *true negatives* ($\#TN$), *false positives* ($\#FP$), and *false negatives* ($\#FN$) as follows:

$$\begin{bmatrix} \#TP & \#FN \\ \#FP & \#TN \end{bmatrix}. \tag{1}$$

This matrix contains all the raw information about the predictions done by a classification model on a given data set. To evaluate the generalization accuracy of a model, it is common to use a *testing data set* which was not used during the learning process of said model. Many synthetic one-dimensional performance indicators can be extracted from a confusion matrix. The performance indicator can be, for example, the precision, the recall, the F-score ... When different kinds of errors are not assumed to be equal, in association with a 2×2 cost matrix, cost-sensitive performance indicators [3] can also be computed from the confusion matrix. The choice of the suitable performance indicator is directly linked to the objective of the learning problem.

Let us assume that we have two models and we want to select the best one according to a given indicator. Performance indicators are scalar numbers computed from the confusion matrix. Assume that the F-scores of the two models are respectively 0.6 and 0.65. Classical methods give no information about the confidence we can have about these values. As performance indicators are intrinsically generated by a random process, we can't be sure that the model with the highest indicator (i.e. 0.65) is really the best one. We would like to find a way to quantify this uncertainty. One of the known techniques to estimate the variability of an indicator is the *bootstrap* method [2].

Note that we are not trying to estimate the generalization accuracy of a whole *learning machine algorithm*.[1] Instead, we study the generalization accuracy of a given model. In this paper, we assume that we don't have access to the *training set* or to the *testing set*. We only have access to the confusion matrix and, on this basis, we try to deduce some properties of the underlying distribution of the model's performance indicators.

In this paper, we propose to use Bayesian techniques [5] on the confusion matrix. For this, we will assume that the values in the confusion matrix are coming from a multinomial distribution [4]. In this parametric context, Bayesian techniques allow us to take into account the intrinsic variability of the unknown parameters of the multinomial distribution. The variability of these parameters will be measured by a probability function, and we will see that this function can be modelized by a *Dirichlet* distribution [4]. This *Dirichlet* distribution can be used in the multinomial distribution to obtain information about the distribution of the values in the confusion matrix. As the performance indicators come from the confusion matrix, it gives us information about the indicators' distribution. This makes it possible to compute metrics about the uncertainty that concerns any performance indicator. The use of a Bayesian framework also allows us to inject *a priori* knowledge in the confusion matrix. We will see that injecting *prior* knowledge can have a positive impact on the *a posteriori* distribution associated to an indicator, which is especially true when the number of measures in the confusion matrix is low.

To the best of the author's knowledge, [6] is the only paper which proposes to use Bayesian methods to assess the confidence of indicators computed from the confusion matrix. However, in [6], the framework may not be applicable to arbitrary performance measures. In our paper, we extend the work by using Dirichlet distributions, which gives us a way to generalize the method for any performance indicator extracted from a confusion

[1] As is generally sought with cross-validation methods.

matrix. We study the impact on the *a posteriori* distribution when *a priori* knowledge is injected, and we also compare our method with the bootstrap techniques.

This paper is structured as follows : The learning problem is first formalized in Section 2. We start Section 3 by assuming that the data in the confusion matrix are generated from a multinomial distribution, and we end this section by showing how to compute the *a posteriori* distribution of any performance indicator generated from this confusion matrix. We illustrate our method on a simple synthetic example in Section 4. By using Bayesian techniques, we have the possibility of injecting *prior* knowledge in the confusion matrix. This is the topic of Section 5. In Section 6, the bootstrap method is theoretically compared to our method based on Bayesian techniques. Section 7 contains experimental results on real and synthetic data sets. We end by the conclusions in Section 8.

2 Formalization of the learning problem

Let $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N_\mathcal{D}}$ be a learning data set with $N_\mathcal{D}$ independent samples. We assume that all the samples follow the same unknown joint distribution $(X_i, Y_i) \sim F(X, Y)$ where $X \in \mathcal{X} \subset \mathbb{R}^d$ and $Y \in \mathcal{Y} = \{\vartheta_0, \vartheta_1, \vartheta_2, \ldots, \vartheta_{|\mathcal{Y}|-1}\}$. It is common to call X the *input* and Y the *output*. As the support of Y contains nominal values, we are in a classification learning problem. For simplicity's sake, we will assume that we are in a binary classification task ($|\mathcal{Y}| = 2$). This means that the set $\mathcal{Y}$ only contains the two following elements $\{\vartheta_0, \vartheta_1\}$. As we will see, the results can easily be generalized for multiclass tasks.

Following the standard machine learning setup [7], the goal is to use $\mathcal{D}$ to learn a classifier model $h : \mathcal{X} \to \mathcal{Y}$. For a new input point $x \in \mathcal{X}$, the classifier returns a prediction $\hat{y} = h(x) \in \mathcal{Y}$. If we associate the value 0 to ϑ_0 and the value 1 to ϑ_1, then this definition includes the case $\hat{y} = h(x) = I\,(g(x) > \gamma) \in \{0, 1\}$ where I is the *indicator* function, $g : \mathcal{X} \to [0, 1] \subset \mathbb{R}$ is a model which outputs real values and $\gamma \in [0, 1]$ is a *threshold*. In this setting, g is a model that returns a confidence score for x to be in class 1. A threshold γ is used to decide whether x belongs to class 1 or 0.

Let $\mathcal{T} = \{(X_i, Y_i)\}_{i=1}^{N_\mathcal{T}}$ be a testing data set with $N_\mathcal{T}$ independent samples following the same unknown distribution $F(X, Y)$. In order to quantify the quality of the predictions made by h on the samples in $\mathcal{T}$, we define the following loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \{TP, TN, FP, FN\}$. Let $y \in \{\vartheta_0, \vartheta_1\}$ be the true class and $\hat{y} \in \{\vartheta_0, \vartheta_1\}$ the prediction. By convention, we define the mapping of the Δ function as follows:

- if $y = \vartheta_1$ and $\hat{y} = \vartheta_1$, then $\Delta(y, \hat{y}) = TP$
- if $y = \vartheta_0$ and $\hat{y} = \vartheta_0$, then $\Delta(y, \hat{y}) = TN$
- if $y = \vartheta_0$ and $\hat{y} = \vartheta_1$, then $\Delta(y, \hat{y}) = FP$
- if $y = \vartheta_1$ and $\hat{y} = \vartheta_0$, then $\Delta(y, \hat{y}) = FN$

To simplify the notation, let $\Delta_i = \Delta(y_i, \hat{y}_i)$. The vector $\mathcal{M} = \left(\Delta_1, \ldots, \Delta_{N_\mathcal{T}}\right)$ contains the values of the loss function computed on the testing set. We count in the vector $\mathcal{V} = (\#TP, \#TN, \#FP, \#FN) \in \mathbb{N}^4$ the number of times we obtain from Δ a TP, a TN, a FP or a FN where $\#TP = \sum_i I(\Delta_i = TP)$, $\#TN = \sum_i I(\Delta_i = TP)$, ... Note that by definition, the sum of the elements in $\mathcal{V}$ equals $N_\mathcal{T}$.

Given a classifier h and its results on $\mathcal{T}$, the vector $\mathcal{V}$ contains all the raw information about the classifier's predictions on the testing set. As in (1), it is common to present this vector in a 2×2 matrix form. In this case, it is called the *confusion matrix*.

A one-dimensional performance indicator $\mathcal{I} : \mathbb{N}^4 \rightarrow \mathbb{R}$ is a function that maps a vector $\mathcal{V}$ into a score used to assess a classifier's prediction performance. Many one-dimensional performance indicators can be computed from $\mathcal{V}$. Common indicators are for example:

- accuracy $= \frac{\#TP+\#TN}{\#TP+\#FP+\#FN+\#TN}$
- precision $= \frac{\#TP}{\#TP+\#FP}$
- recall $= \frac{\#TP}{\#TP+\#FN}$
- F_β-score $= (1+\beta^2)\frac{\text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}}$; where $\beta \in \mathbb{R}^+$
- G-score $= \sqrt{\text{precision} \cdot \text{recall}}$
- MCC $= \frac{\#TP\cdot\#TN-\#FP\cdot\#FN}{\sqrt{(\#TP+\#FP)(\#TP+\#FN)(\#TN+\#FP)(\#TN+\#FN)}}$

The performance indicator function $\mathcal{I}$ can be used to compare different classifiers. We usually take the model with the best performance.

3 A posteriori probability distribution of a performance indicator

The testing set $\mathcal{T}$ contains $N_\mathcal{T}$ independent and identically distributed random samples. As the classifier h is tested on $\mathcal{T}$, the scalar $\mathcal{I}(\mathcal{V})$ is itself a random variable from which we want to deduce some properties of the underlying distribution. To do that, we propose to adopt a Bayesian framework on the confusion matrix $\mathcal{V}$. We will first consider that the values in the confusion matrix are coming from a multinomial distribution with unknown parameters. The use of a Bayesian framework gives us the opportunity to assume that the unknown parameters are generated from a *Dirichlet* distribution. We can add *a priori* knowledge in the *Dirichlet* probability function to determine an *a posteriori* probability distribution of the unknown parameters. This distribution can then be used to compute an *a posteriori* of the one-dimensional performance indicators acquired from the confusion matrix.

The output of the loss function Δ can be seen as the result of a random experiment with $\{TP, TN, FP, FN\}$ as support set. It is a generalization of a *Bernoulli trial* in which, rather than only two outputs for each trial, we have four. As the data in $\mathcal{T}$ are independent and identically distributed random variables, we know that the $N_\mathcal{T}$ outputs of Δ are also independent and identically distributed. After a series of $N_\mathcal{T}$ independent random trials, the vector $\mathcal{V}$ can be interpreted as a random vector where the elements $\#TP$, $\#TN$, $\#FP$ and $\#FN$ contain the number of times that we observe TP, TN, FP and FN, respectively. The *binomial distribution* is the discrete probability distribution of the number of successes in a sequence of independent *Bernoulli trials*. The *multinomial distribution* is a generalization of a *binomial distribution* when there are more than two possible outputs at each trial. As the vector $\mathcal{V}$ counts the number of times that we observe TP, TN, FP and FN, the vector $\mathcal{V}$ follows a *multinomial distribution* where there are four possible outputs at each of the $N_\mathcal{T}$ independent trials.

In this context, the vector $\mathcal{V}$ follows a *multinomial distribution*

$$\mathcal{V} \sim \text{Mult}(N_\mathcal{T}, \theta)$$

with the following probability mass function:

$$P(\mathcal{V} = v) = \frac{N_\mathcal{T}!}{v_1! \cdot v_2! \cdot v_3! \cdot v_4!} \cdot \theta_{tp}^{v_1} \cdot \theta_{tn}^{v_2} \cdot \theta_{fp}^{v_3} \cdot \theta_{fn}^{v_4} \cdot I\left(\sum_{i=1}^{4} v_i = N_\mathcal{T}\right) \tag{2}$$

where $\theta = (\theta_{tp}, \theta_{tn}, \theta_{fp}, \theta_{fn}) \in S_\theta \subset \mathbb{R}^4$ are the unknown parameters of the multinomial distribution and $v = (v_1, v_2, v_3, v_4)$ is a realization of $\mathcal{V}$ with four numbers in $\mathbb{N}$. The set S_θ is called the *probability simplex* and contains all the possible values of θ:

$$S_\theta = \{\ \theta \mid \theta_{tp} \geq 0, \theta_{tn} \geq 0, \theta_{fp} \geq 0, \theta_{fn} \geq 0 \text{ and } \theta_{tp} + \theta_{tn} + \theta_{fp} + \theta_{fn} = 1\}.$$

In (2), it is assumed that θ is a vector with four *fixed* unknown parameters. Adopting a Bayesian point of view, we can consider that θ is a realization of an unknown random variable Θ. In this Bayesian setting, the left part of (2) becomes $P(\mathcal{V} = v|\Theta = \theta)$.

The Bayes rule and the law of total probability tell us that

$$\begin{aligned} f_{\Theta|\mathcal{V}}(\theta|v) &= \frac{P(\mathcal{V} = v|\Theta = \theta) \cdot f_\Theta(\theta)}{P(\mathcal{V} = v)} \\ &= \frac{P(\mathcal{V} = v|\Theta = \theta) \cdot f_\Theta(\theta)}{\int_{S_\theta} P(\mathcal{V} = v|\Theta = \theta) \cdot f_\Theta(\theta) \cdot d\theta} \end{aligned} \tag{3}$$

where $f_{\Theta|\mathcal{V}}(\theta|v)$ is the conditional density function of Θ, given a confusion matrix v.

Note that in (3), the denominator is there to ensure that $\int_{S_\theta} f_{\Theta|\mathcal{V}}(\theta|v)d\theta = 1$. Consequently, we only have to evaluate the numerator and to normalize the results so that the integral at the end equals 1. In this setting, (3) becomes

$$\underbrace{f_{\Theta|\mathcal{V}}(\theta|v)}_{\text{A posteriori}} \propto \underbrace{P(\mathcal{V} = v|\Theta = \theta)}_{\text{Likelihood}} \cdot \underbrace{f_\Theta(\theta)}_{\text{A priori}}. \tag{4}$$

Thanks to the Bayes rule, which allows us to interpret the parameter vector θ as a realization of a random variable, (4) gives the conditional density function of this variable Θ. This conditional density is proportional to the product of two terms. The *likelihood* returns a plausibility score that the values in the confusion matrix are generated from a multinomial distribution with parameters θ. The second term is the *a priori* distribution and, as we will see, it allows us to inject *prior* knowledge about the accuracy of the classifier model h. The *a posteriori* probability $f_{\Theta|\mathcal{V}}(\theta|v)$ is a compromise between the *a priori* and the *likelihood*.

The Dirichlet distribution is the conjugate distribution of the multinomial distribution [5]. It ensures that if the *likelihood* follows a multinomial function and the *a priori* follows a Dirichlet function, then the *a posteriori* will also follow a Dirichlet function. The Dirichlet distribution is commonly used in Bayesian statistics to model the parameters of a multinomial distribution. In this context, we say that the vector Θ follows a Dirichlet distribution

$$\Theta \sim \text{Dir}(\alpha) = \text{Dir}((\alpha_1, \alpha_2, \alpha_3, \alpha_4))$$

with the following density function:

$$\begin{aligned} f_\Theta(\theta) &= \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)} \cdot \theta_{tp}^{\alpha_1 - 1} \cdot \theta_{tn}^{\alpha_2 - 1} \cdot \theta_{fp}^{\alpha_3 - 1} \cdot \theta_{fn}^{\alpha_4 - 1} \cdot I(\theta \in S_\theta) \\ &\propto \theta_{tp}^{\alpha_1 - 1} \cdot \theta_{tn}^{\alpha_2 - 1} \cdot \theta_{fp}^{\alpha_3 - 1} \cdot \theta_{fn}^{\alpha_4 - 1} \cdot I(\theta \in S_\theta) \end{aligned} \tag{5}$$

where the variables $\alpha_i \in \mathbb{R}_0^+$ are the parameters of the Dirichlet distribution and where $\Gamma(\cdot)$ is the *gamma function*. The first term $\Gamma\left(\sum_i \alpha_i\right)/\prod_i \Gamma(\alpha_i)$ in the previous equation can be omitted because it is only there to ensure that the integral of $f_\Theta(\theta)$ on S_θ equals 1.

The (5) defines the *a priori* distribution of the parameters in θ, and these parameters are used in (2) to define the probability distribution of the values in the confusion matrix. *Prior*

knowledge can be injected in the *a priori* distribution via $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. This topic will be covered in Section 5.

If in (4), the *likelihood* and the *a priori* are respectively replaced by (2) and (5), we have

$$f_{\Theta|\mathcal{V}}(\theta|v) \propto \theta_{tp}^{v_1+\alpha_1-1} \cdot \theta_{tn}^{v_2+\alpha_2-1} \cdot \theta_{fp}^{v_3+\alpha_3-1} \cdot \theta_{fn}^{v_4+\alpha_4-1} \cdot I\,(\theta \in S_\theta)\,.$$

This is the *a posteriori* probability density function of the unknown parameters. We can identify that, given the confusion matrix v and the prior α, this *a posteriori* density function of Θ follows a Dirichlet distribution

$$\Theta|\omega \sim \text{Dir}((v_1 + \alpha_1, v_2 + \alpha_2, v_3 + \alpha_3, v_4 + \alpha_4)) = \text{Dir}(\omega) \tag{6}$$

where $\omega = (\omega_1, \omega_2, \omega_3, \omega_4) = (v_1 + \alpha_1, v_2 + \alpha_2, v_3 + \alpha_3, v_4 + \alpha_4)$ are the parameters of the *a posteriori* density function.

In (6), we have an analytical definition of the *a posteriori* distribution of the unknown parameters. In the following, we will see how we can use this definition to compute the *a posteriori* distribution of $\mathcal{V}$, given the observed confusion matrix v and the *prior* knowledge α. Let $\tilde{v}$ be an arbitrary new confusion matrix. Before the data in v are considered, the probability to observe this unknown matrix $\tilde{v}$ is

$$P(\mathcal{V} = \tilde{v}) = \int_{S_\theta} P(\mathcal{V} = \tilde{v}|\Theta = \theta)\; f_\Theta(\theta)\; d\theta = E_{f_\Theta}\left[P(\mathcal{V} = \tilde{v}|\Theta)\right].$$

It is often called the *a priori predictive distribution*. After the matrix v has been observed, we can compute an *a posteriori predictive distribution* by replacing $f_\Theta(\theta)$ with the *a posteriori* distribution $f_{\Theta|\mathcal{V}}(\theta|v)$ as follows:

$$P(\mathcal{V} = \tilde{v}|v) = \int_{S_\theta} P(\mathcal{V} = \tilde{v}|\Theta = \theta)\; f_{\Theta|\mathcal{V}}(\theta|v)\; d\theta = E_{f_{\Theta|\mathcal{V}}}\left[P(\mathcal{V} = \tilde{v}|\Theta)\right]. \tag{7}$$

The distribution of $\mathcal{V}|v$ synthesizes all the relevant information we can have about the confusion matrix. It shows that the *a posteriori* predictive function is the expectation of the conditional probability $P(\mathcal{V} = \tilde{v}|\Theta = \theta)$ over the *a posteriori* distribution of Θ. In (7), $P(\mathcal{V} = \tilde{v}|\Theta = \theta)$ is obtained by the multinomial model for a given value of the parameters and $f_{\Theta|\mathcal{V}}(\theta|v)$ is the *a posteriori* distribution of said parameters.

A performance indicator $\mathcal{I}$ is a function that maps the values of a confusion matrix into real number space. Let $G = \mathcal{I}(\mathcal{V}|v)$ be a random variable associated with the values taken by the performance indicator function. The density function of G can theoretically be computed by a reparameterization [10] of the probability function in (7). But most of the time, the analytical calculation of the distribution of G may be very complex, or even impossible. Instead, we propose to estimate this distribution by the following Monte Carlo simulation process, described in algorithm 1.

Algorithm 1 Generating random samples from G by Monte Carlo

```
1: function MC-SAMPLER(I, α, v, M)
2:     for m ← 1 to M do
3:         Sample θ_m from f_{Θ|V}(θ|ω)          ▷ See equation (6)
4:         Sample ṽ_m from P(V = ṽ|Θ = θ_m)      ▷ See equation (2)
5:         g_m ← I(ṽ_m)
6:     end for
7:     return g = (g_1, ..., g_M)
8: end function
```

Input in this algorithm are : (i) a performance indicator function $\mathcal{I}$, (ii) a vector α with *prior* knowledge, (iii) a vector v with the values of the observed confusion matrix and (iv) the number M of samples we want to generate. The main loop appears between lines 2 and 6. In line 3, a sample θ_m is generated from a Dirichlet[2] distribution where ω is a vector containing the sum of the vectors α and v. This sample θ_m is used in line 4 to extract a sample $\tilde{v}_m$ from a multinomial distribution. In line 5, $\tilde{v}_m$ is injected in the performance indicator function $\mathcal{I}$ to compute g_m. Note that if a correlation study must be done between the indicators, it should be at this line that the other performance indicators are computed on the same $\tilde{v}_m$. In Section 4, we will use this technique to evaluate the correlation between the precision and recall on a synthetic example. At line 7, the algorithm returns a vector with M samples from G. These M samples can then be used to estimate any statistics about the distribution of the indicator, such as the mean, variance, skewness, quantiles...

Note that to generate the M samples from the random variable associated with the performance indicator G, we don't need to know the learning set $\mathcal{D}$ or the testing set $\mathcal{T}$. Since the total number of samples has to be known in order to obtain a multinomial distribution, we only need one observed realization v of the random vector $\mathcal{V}$ associated to the confusion matrix.

4 Example

To illustrate our method, let us consider an example with two classifiers A and B, respectively producing the two following confusion matrices on the same testing data set $\mathcal{T}$ where $N_{\mathcal{T}} = 145$:

$$\begin{bmatrix} 65 & 15 \\ 35 & 30 \end{bmatrix} \Rightarrow v^A = (65, 30, 35, 15)$$
$$\begin{bmatrix} 50 & 30 \\ 30 & 35 \end{bmatrix} \Rightarrow v^B = (50, 35, 30, 30).$$

The superscript above v indicates the model from which the confusion matrix is coming from. If $\mathcal{I}$ is the Matthews Correlation Coefficient (MCC), then in our example we have $\mathcal{I}(v^A) = 0.2946$ and $\mathcal{I}(v^B) = 0.1635$. Based on this criteria, it seems that the classifier A outperforms the other one. But we don't have any information to decide if this is really the case or due to chance.

Let us now adopt the Bayesian framework with $\alpha = (0, 0, 0, 0)$. From (6), the *a posteriori* distributions of the unknown parameters θ are:

$$\alpha = (0, 0, 0, 0) \Longrightarrow \begin{cases} (\Theta|v^A, \alpha) \sim \text{Dir}(w = (65, 30, 35, 15)) \\ (\Theta|v^B, \alpha) \sim \text{Dir}(w = (50, 35, 30, 30)) \end{cases}$$

By using the algorithm 1, let $g^i = (g^i_1, \ldots, g^i_M)$ be a set with $M = 1,000,000$ random samples generated from $\mathcal{I}(\mathcal{V}|v^i)$ where $i \in \{A, B\}$.

[2]To generate random samples from a Dirichlet distribution, we can use the following property : $\forall i \in \{1, \ldots, N\}$, $X_i \sim \text{gamma}(\varphi_i, 1) \Rightarrow (X_1/X_\Sigma, \ldots, X_N/X_\Sigma) \sim \text{Dir}(\varphi_1, \ldots, \varphi_N)$ where $X_\Sigma = \sum_i X_i$.

In the left part of Fig. 1, we used g^A and g^B to show the estimated distributions of $\mathcal{I}(\mathcal{V}|v^A)$ and $\mathcal{I}(\mathcal{V}|v^B)$. As expected, model A seems better than model B. Adopting the Bayesian point of view, g^A and g^B can also be used to infer other quantities such as:

$$P(\mathcal{I}(\mathcal{V}|v^B) > 0) \approx \frac{1}{M}\sum_{m=1}^{M} I(g_m^B > 0) \approx 0.92$$

$$P(\mathcal{I}(\mathcal{V}|v^A) > \mathcal{I}(\mathcal{V}|v^B)) \approx \frac{1}{M}\sum_{m=1}^{M} I\left(g_m^A > g_m^B\right) \approx 0.79$$

$$CI_{0.95}(\mathcal{I}(\mathcal{V}|v^B)) \approx \text{hpd}(g_m^B) \approx [-0.07, 0.39].$$

where CI stands for *credible interval*. The first value means that we have 92% plausibility that the MCC of model B is positive. The second value shows that we have a high degree of plausibility that model A really outperforms model B in terms of MCC. The last equation gives a 95% credible interval for the MCC of model B. To compute the credible interval, we extract the *highest posterior density*(hpd) [5]. The hpd is a more robust way to extract intervals when the distribution is asymmetrical.

It is also possible to study the joint distribution of more than one performance indicator. As already mentioned, in line 5 of algorithm 1, we can compute the value of two indicators instead of only one. In the right part of Fig. 1, we display 1,000 samples of the recall and precision generated from the confusion matrix of model A. Many statistics can be extracted from this scatter plot. We can, for example, infer that the Pearson correlation has a 95% chance of being in the interval [0.21, 0.32].

5 Adding prior knowledge in the distribution

As already mentioned, the Bayesian framework allows us to inject *prior* knowledge into the *a posteriori*. Two distinct situations can occur : either we are in a situation of complete

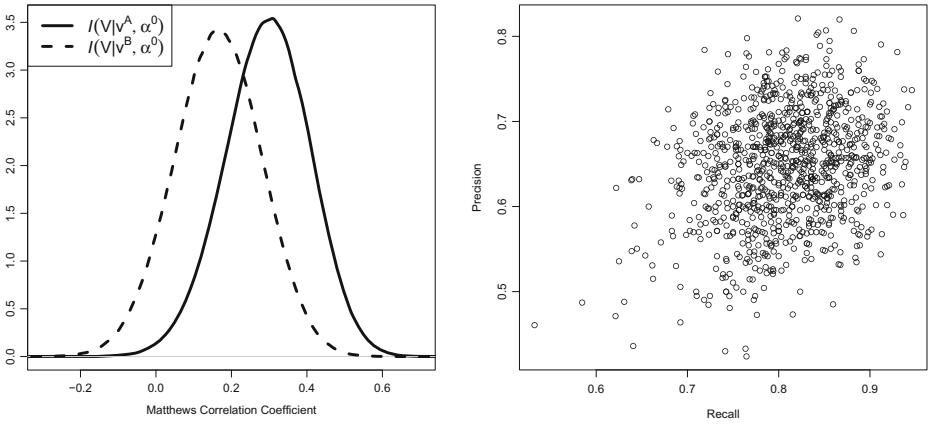

Fig. 1 Left: We compare the estimated distributions $\mathcal{I}(\mathcal{V}|v^A)$ and $\mathcal{I}(\mathcal{V}|v^B)$ and we see that the distribution of model A's MCC is better. Right: Empirical distribution of the precision and recall

uncertainty about the distribution, or we have some *prior* knowledge to inject in the *a posteriori* distribution.

In the first case, the Dirichlet probability function $f_\Theta(\theta)$ achieves maximum entropy when $\alpha = (1, 1, 1, 1)$. From (5), we can see that the *a priori* distribution $f_\Theta(\theta) \propto I(\theta \in S_\theta)$ is then an uniform distribution on the probability simplex S_θ. This means that if there is no *prior* knowledge to inject in the *a posteriori*, we can just switch all the α values to one.

In (6), the values $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and $v = (v_1, v_2, v_3, v_4)$ are coming from the *a priori* knowledge and the confusion matrix, respectively. The vector ω with the parameters of the distribution $f_{\Theta|\mathcal{V}}(\theta|v)$ is the sum of α and v. The duality between α and v in (6) suggests that this *prior* conveys the same information as a pilot fake test where the model would have been tested four times, with $\alpha = (1, 1, 1, 1)$ containing the number of times TP, TN, FP and FN appear during this test. So, by using this *prior*, we inject a small bias in the *a posteriori* distribution. Therefore, someone choosing a uniform *prior* is not in a state of complete ignorance as (s)he discards the possibility of having a value equal to zero in the confusion matrix. This suggests that we take as *prior* an *improper*[3] Dirichlet distribution where $\alpha = (0, 0, 0, 0)$, as it is equivalent to the absence of a pilot study [5]. This α has the advantage of injecting no bias in the *a posteriori*, but the drawback is that by using this improper *a priori* distribution $f_\Theta(\theta)$, we are not guaranteed to obtain a proper *a posteriori* distribution $f_{\Theta|\mathcal{V}}(\theta|v)$. If one of the elements in the observed confusion matrix v is a zero, then the *a posteriori* distribution will also be improper.

Let us take from Section 4 the same confusion matrix $v^A = (65, 30, 35, 15)$, and let us compare the impact on the *a posteriori* where $\alpha = (0, 0, 0, 0)$ or $\alpha = (1, 1, 1, 1)$ are used as *prior*. From (6), the *a posteriori* distribution of the unknown parameters θ are:

$$\begin{cases} \alpha^0 = (0, 0, 0, 0) \Longrightarrow (\Theta|v^A, \alpha^0) \sim \text{Dir}(w = (65, 30, 35, 15)) \\ \alpha^1 = (1, 1, 1, 1) \Longrightarrow (\Theta|v^A, \alpha^1) \sim \text{Dir}(w = (66, 31, 36, 16)) \end{cases}$$

Algorithm 1 is used to generate $M = 1,000,000$ samples from both random variables $\mathcal{I}(\mathcal{V}|v^A, \alpha^0)$ and $\mathcal{I}(\mathcal{V}|v^A, \alpha^1)$. In the left part of Fig. 2, we used g^{A0} and g^{A1} to display the curve of the two distributions. As we can see, there is a minor impact when $\alpha^1 = (1, 1, 1, 1)$ is injected as *prior* knowledge.

A scientist is often not in a state of perfect ignorance with respect to the performance of the models that (s)he is using. This *prior* knowledge can come from the scientist's experience or from previous studies done in the same context. The *prior* knowledge can be injected directly via $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. For example, as α_1 and α_2 contain the *prior* information about $\#TP$ and $\#TN$ respectively and if prior studies show that we should have an accurate model, we can use $\alpha = (\alpha_1, \alpha_2, 1, 1)$ and set α_1, α_2 to a high value. This is a very approximate and imprecise way of adding prior knowledge.

Note that adding *prior* knowledge could have a negative impact if the knowledge that was injected was wrong. In this case, a bias would be added in the *a posteriori* distribution, but its impact would decline when the number of elements in the testing set $N_\mathcal{T}$ increases.

Injecting *prior* knowledge directly via $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ is sometimes difficult, even for experts in the application area. Therefore, it is often easier to compute the values in

[3] Roughly speaking, *improper* means that we lose the fundamental property of a probability density function that its integral must be one.

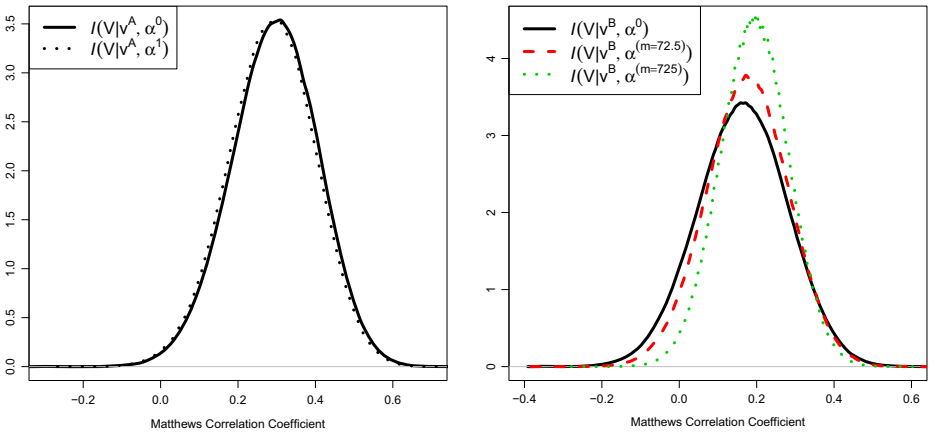

Fig. 2 Left: we compare the distributions of $\mathcal{I}(\mathcal{V}|v^A, \alpha^0)$ and $\mathcal{I}(\mathcal{V}|v^A, \alpha^1)$. We observe that the impact of using α^1 is relatively low. Right: Evolution of the distribution when prior knowledge is injected in the distribution

α in an indirect way. Rather than information on α, the scientist often has more knowledge/intuition about the values of some classical accuracy measures. If we define $p = \#TP/(\#TP + \#FP)$, $r = \#TP/(\#TP + \#FP)$ and $a = (\#TP + \#TN)/(\#TP + \#FP + \#FN + \#TN)$ as the precision, the recall and the accuracy respectively, then we can see that

$$\begin{cases} \alpha_1 = \#TP &= \frac{m(1-a)rp}{p+r-2rp} \\ \alpha_2 = \#TN &= am - \#TP \\ \alpha_3 = \#FP &= \frac{m(1-a)r(1-p)}{p+r-2rp} \\ \alpha_4 = \#FN &= \frac{m(1-a)(1-r)p}{p+r-2rp} \end{cases} \tag{8}$$

where $m = \#TP + \#FP + \#FN + \#TN > 0$ and $r^{-1} + p^{-1} - a^{-1} > 1$. The last inequality is there to ensure that $\#TN > 0$.

Let us now evaluate the impact on the *a posteriori* distribution when we inject *prior* knowledge in the confusion matrix $v^B = (50, 35, 30, 30)$. Assume that a scientist can make the following assumption about model B : $precision = 0.6$, $recall = 0.65$ and $accuracy = 0.6$. From (8), we have:

$$\begin{cases} \#TP &= 0.3319149 \times m \\ \#TN &= 0.2680851 \times m \\ \#FP &= 0.2212766 \times m \\ \#FN &= 0.1787234 \times m \end{cases}$$

where m is a parameter used to put a weight on the *prior*. We tested the two following values for $m : m = N_{\mathcal{T}}/2 = 72.5$ and $m = N_{\mathcal{T}} * 5 = 725$, and obtained the following distribution for the parameters:

$$(\Theta|v^B, \alpha^0) \sim \text{Dir}(w = (50, 35, 30, 30))$$

$$(\Theta|v^B, \alpha^{m=72.5}) \sim \text{Dir}(w = (74.063, 54.436, 46.042, 42.957))$$

$$(\Theta|v^B, \alpha^{m=725}) \sim \text{Dir}(w = (290.638, 229.361, 190.425, 159.574))$$

where, for example, 74.063 is obtained by doing $50+0.3319149 \times 72.5$. The right part of Fig. 2 shows the impact when *prior* knowledge is added. We see that adding *prior* knowledge reduces the variance of the distribution, and that this reduction amplifies when m increases.

6 Bootstrap and Bayesian approach

The bootstrap method is another technique that can be used to approximate the distribution of $\mathcal{I}(\mathcal{V}|v)$ in an easy and accurate way. Note that the bootstrap is often used in machine learning to estimate the generalization accuracy of a learning algorithm by bootstrapping the training set. But this is not what we will do here. As we assume in this paper that we don't have access to the training set, we will do the bootstrap directly on the confusion matrix.

Given a testing set $\mathcal{T}$ and a model h, the vector $\mathcal{M} = \left(\Delta_1, \ldots, \Delta_{N_{\mathcal{T}}}\right)$ contains the $N_{\mathcal{T}}$ values of the loss function computed on $\mathcal{T}$ and the vector $\mathcal{V}$ is an aggregation of $\mathcal{M}$ where we count in $\#TP$, $\#TN$, $\#FP$ and $\#FN$ the number of times we have observed TP, TN, FP and FN, respectively. For bootstrap, B samples $\left\{\mathcal{M}^{*(1)}, \ldots, \mathcal{M}^{*(B)}\right\}$ are generated where $\mathcal{M}^{*(b)} = \left(\Delta_1^{*(b)}, \ldots, \Delta_{N_{\mathcal{T}}}^{*(b)}\right)$ is obtained from $\mathcal{M}$ by doing $N_{\mathcal{T}}$ random samplings with replacement. After aggregating $\mathcal{M}^{*(b)}$ in $\mathcal{V}^{*(b)}$, the B vectors $\mathcal{V}^{*(b)}$ are used to compute the B values $\mathcal{I}(\mathcal{V}^{*(b)})$. These B values can then be used to infer properties about the distribution of the statistic $\mathcal{I}(\mathcal{V}|v)$. The following algorithm formalizes the process.

Algorithm 2 Generating random samples from G by bootstrap

```
1: function BOOT-SAMPLER(I, M, B)
2:     for b ← 1 to B do
3:         M*(b) ← Random sampling with replacement in M
4:         V*(b) ← Aggregate(M*(b))
5:         g*(b) ← I(V*(b))
6:     end for
7:     return g* = (g*(1), ..., g*(B))
8: end function
```

In Section 3, we have assumed that the output of the loss function Δ could be interpreted as the result of a random experiment with $\{TP, TN, FP, FN\}$ as support set. Let $\mathcal{P} = \{P(\Delta = k)\}$ with $k \in \{TP, TN, FP, FN\}$ be a set with the four unknown probabilities associated to the four values in support of the loss function Δ. Given the $N_{\mathcal{T}}$ observed values of the loss function, the maximum likelihood estimator of these probabilities is $\widehat{\mathcal{P}} = \left\{\widehat{P}(\Delta = k)\right\}$ where $\widehat{P}(\Delta = k) = \#k/N_{\mathcal{T}}$. In the bootstrap method, $\Delta^{*(b)}$ is obtained by doing a random sampling with replacement in $\mathcal{M}$, which is equivalent to doing a sampling

in $\{TP, TN, FP, FN\}$ with probabilities $\widehat{\mathcal{P}}$. As the random vector $\mathcal{V}^{*(b)}$ is a counting aggregation of $\mathcal{M}^{*(b)}$, the random vector $\mathcal{V}^{*(b)}$ follows a multinomial distribution

$$\mathcal{V}^* \sim \text{Mult}(N_{\mathcal{T}}, \widehat{\mathcal{P}}).$$

Compared to the Bayesian method studied in this paper, the previous equation shows that the bootstrap method on $\mathcal{M}$ produces samples of the confusion matrix that follow a multinomial distribution with four *fixed* parameters. In the Bayesian approach, we assume that these parameters themselves are uncertain and follow a Dirichlet distribution. Roughly speaking, we can say that, in bootstrap, these parameters follow a Dirac distribution.

As a consequence, by replacing the fixed parameters with random values, we can see that the Bayesian approach has a tendency to have more variability in the distribution of $\mathcal{V}$. Let $\mathcal{I}(\mathcal{V}|v)$ and $\mathcal{I}(\mathcal{V}^*)$ be the random variables for the performance indicator functions obtained by the Bayesian approach and the bootstrap approach, respectively. By the law of total variance, we have

$$\begin{aligned} V\left[\mathcal{I}(\mathcal{V}^*)\right] &= E_\Theta\left[V\left[\mathcal{I}(\mathcal{V}^*)|\Theta\right]\right] + V_\Theta\left[E\left[\mathcal{I}(\mathcal{V}^*)|\Theta\right]\right] \\ V\left[\mathcal{I}(\mathcal{V}|v)\right] &= E_\Theta\left[V\left[\mathcal{I}(\mathcal{V}|v)|\Theta\right]\right] + V_\Theta\left[E\left[\mathcal{I}(\mathcal{V}|v)|\Theta\right]\right]. \end{aligned}$$

As there is no variability in the parameters for bootstrap, we have $V_\Theta\left[E\left[\mathcal{I}(\mathcal{V}^*)|\Theta\right]\right] = 0$ and therefore $V_\Theta\left[E\left[\mathcal{I}(\mathcal{V}|v)|\Theta\right]\right] > V_\Theta\left[E\left[\mathcal{I}(\mathcal{V}^*)|\Theta\right]\right]$. As a consequence, $V\left[\mathcal{I}(\mathcal{V}|v)\right]$ tends to be higher than $V\left[\mathcal{I}(\mathcal{V}^*)\right]$, and accordingly, the variability of $\mathcal{I}(\mathcal{V}|v)$ obtained by the Bayesian approach is often higher than the variability of $\mathcal{I}(\mathcal{V}^*)$ obtained by the bootstrap approach.

We will now compare our strategy with the bootstrap method, and see what happens when the number of samples in the testing set $\mathcal{T}$ increases. Let $e(\lambda) = \lambda v^A = (\lambda 65, \lambda 30, \lambda 35, \lambda 15)$ be a vector where $\lambda > 0$. To simulate an increase of $N_{\mathcal{T}}$, we will take the values of $e(\lambda)$ where $\lambda \in (1, 2, 3, \ldots, 500)$. For each value of λ, we generate 1,000,000 values of $g^{e(\lambda)}$ with algorithm 1 where α^0 is used as *prior*. The same number of values of $g^{*e(\lambda)}$ is generated by the bootstrap method described in algorithm 2. To compare our strategy with the bootstrap method, we will compare the distributions of $g^{e(\lambda)}$ and $g^{*e(\lambda)}$ when λ increases.

The left part of Fig. 3 shows the evolution of the interquartile distance of $g^{e(\lambda)}$ and $g^{*e(\lambda)}$. Both curves are decreasing, meaning that the variability decreases when more knowledge is available. As expected, we can observe that the variability of the samples generated by the Bayesian method is always higher than that of the samples generated by the bootstrap method. This can be explained by the fact that the Bayesian method takes the variability of the unknown parameters into account, whereas the bootstrap assumes them to be fixed. But we can also notice that the relative distance between the two curves reduces when $N_{\mathcal{T}}$ increases. This is due to the fact that when more information is available, the variability of the unknown parameters declines. Therefore, if $N_{\mathcal{T}}$ becomes too great, the variability of the unknown parameters will become almost null and the two curves will join. Thus, the Bayesian approach tends to produce distributions with higher variance, but one must take into account the ignorance regarding the parameters θ when $N_{\mathcal{T}}$ is small.

The right part of Fig. 3 shows, for each λ, the difference between the empirical mean of the samples in $g^{e(\lambda)}$ and $g^{*e(\lambda)}$. As we can observe, the curve oscillates around the value 0. This means that, on average, the two methods return samples with the same empirical mean. This is confirmed by the fact that a t-test cannot exclude the null hypothesis that the mean of the 500 values, in this figure, equals 0 (p-val = 0.67).

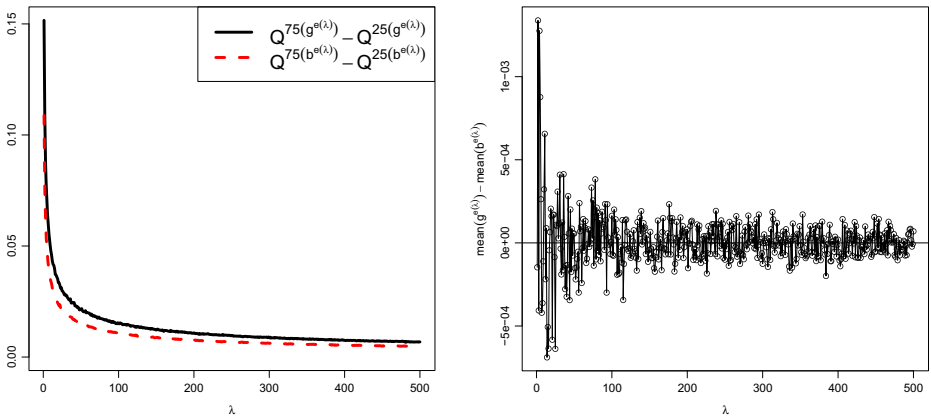

Fig. 3 Left: evolution of the interquartile range of the two methods when the number of samples in the testing set $\mathcal{T}$ increases. Right: Compute, for each value of λ, the difference of the average of the samples generated by bootstrap and the average of the samples generated by our method

7 Experimentation

In this section, we propose two types of experiments. First, we will use real data sets to assess our method's ability to construct accurate credible intervals for different performance indicators. In the second part, we will evaluate our method in a context where the testing sets are sent to the model sequentially, in a series of small chunks. This will allow us to evaluate the possibility of injecting prior knowledge sequentially in the distribution.

7.1 Constructing accurate credible intervals

For the first experiment, we will use random forest models and the data sets described in Table 1. As performance indicators, we will use the accuracy, the G-score and the F_1-score. Algorithm 3 describes the experimental design.

The inputs of the experimental design are a data set $\mathcal{L}$, a performance indicator function $\mathcal{I}$ and a probability δ for the credible interval. In line 2, a variable $score$ is initialized at 0. The value of this variable will be returned at the end of the function and used, as the *coverage probability*, in Table 2. The main loop is between lines 3 – 13 and is repeated 2,000 times. In this loop, the data set is first split in three sub-sets. It is a stratified random sampling, and each sub-set has the same number of elements. In line 5, the learning data set $\mathcal{D}$ is used with a random forest algorithm[4] to generate the model $\mathcal{R}$. This model is used in line 6 to compute the confusion matrix v on the testing data set $\mathcal{T}$. The algorithm 1 described in Section 3 is then used on v to generate a set with 1,000 samples. As some data sets are very small, to avoid introducing too many biases, we use $\alpha^0 = (0, 0, 0, 0)$ as prior in line 7. In line 8, the credible interval $\mathcal{C}$ is computed. To get $\mathcal{C}$, we extract the highest posterior density (hpd) from the samples in g. Now that we have $\mathcal{C}$, the model $\mathcal{R}$ is reused on $\mathcal{S}$ to obtain a new confusion matrix v_s in line 9. Note that as $\mathcal{T}$ and $\mathcal{S}$ are independent, so are the two confusion matrices v and v_s. In lines 10 – 11, we check whether the new performance

[4]We used R's *randomForest* package [8] with the default learning parameters.

Table 1 This table lists the 16 well known real data sets extracted from the *UCI Machine Learning Repository* and used for the experiments. As we are considering binary classification tasks, in the last column (ϑ_1) we have indicated the number of the variable used as output and the value for which the output equals *true*

Data set number	Full data set name	nb obs.	nb var input	ϑ_1
1	Iris	150	4	V5="versicolor"
2	Wine	178	13	V1="1"
3	Blood transfusion service center	748	4	V5="1"
4	Pima indians diabetes	768	8	V9="1"
5	Mammographic mass	961	5	V6="1"
6	Glass identification	214	10	V11="1"
7	Fertility	100	9	V10="O"
8	Ecoli	336	7	V8="cp"
9	Abalone	4177	8	V9="9"
10	Banknote authentication	1372	4	V5="1"
11	Car evaluation	1728	6	V7="acc"
12	Chess (King-Rook vs. King-Pawn)	3196	36	V37="won"
13	Connectionist bench (Sonar)	208	60	V61="M"
14	Contraceptive method choice	1473	9	V10="1"
15	Haberman's survival	306	3	V4="2"
16	Hayes-Roth	132	3	V4="1"

indicator is in the credible interval $\mathcal{C}$. At line 14, the function returns the proportion of time that $\mathcal{I}(v_s)$ was in $\mathcal{C}$.

The experimental results are to be found in Table 2. This table describes the experiment results obtained on the 16 well-known real data sets and the three measures (accuracy, G-score and F_1-score). *Coverage probability* gives the proportion of times that $\mathcal{I}(v_s)$ was in the interval, and *Average length of CI* gives the average length of the 2,000 credible intervals.

As expected, we can see that for each case, when the confidence δ increases, so does the average length of the intervals. Sometimes we also notice that to get a high confidence, the average size of the intervals must remain small. That is the case for the data set number 10, with F_1-score as performance measure. In this case, to have a coverage probability of 0.976, the average size of the intervals must be only 0.03.

The last line of each sub-table in Table 2 provides a *t-based* confidence interval (with $\alpha = 0.05$) of the mean of the 16 measures mentioned above. Concerning the coverage probability, we can see that for each case, the confidence interval contains the target confidence δ. However, in some instances, the coverage probability stays far from the target value. That is the case for the F_1-score with data sets number 6 and 9, where the coverage probability is respectively too small and too high.

Figure 4 displays the density estimation of $\mathcal{I}(v_s)$ and g computed for the 2,000 loops in algorithm 3. $\mathcal{I}(v_s)$ is the value of the performance indicator computed on the independent testing set $\mathcal{S}$. g is computed at line 7 of algorithm 3. g is a vector with 1,000 values, and the 2,000 vectors are put together to compute the density of g in Fig. 4. In Fig. 4, we display the curve of F_1-score for data sets 6 and 13.

Table 2 Experimental results with the Bayesian approach

	Accuracy						
		$\delta = 0.90$		$\delta = 0.95$		$\delta = 0.99$	
Data set number	avg($\mathcal{I}(v)$)	Coverage probability	Average length of CI	Coverage probability	Average length of CI	Coverage probability	Average length of CI
Accuracy							
1	0.95	0.805	0.1058	0.901	0.1333	0.9515	0.1802
2	0.97	0.785	0.0599	0.8015	0.0735	0.8105	0.1048
3	0.76	0.947	0.1234	0.9755	0.1463	0.9975	0.1901
4	0.76	0.9165	0.1216	0.96	0.1444	0.987	0.1877
5	0.82	0.9005	0.0973	0.942	0.1155	0.986	0.15
6	0.99	0.8155	0.0157	0.8225	0.0203	0.8315	0.0297
7	0.87	0.9965	0.2347	0.999	0.2782	1	0.3568
8	0.96	0.8215	0.0773	0.897	0.0924	0.97	0.1221
9	0.83	1	0.0464	1	0.0551	1	0.0715
10	0.99	0.8305	0.0209	0.917	0.0247	0.9705	0.033
11	0.91	0.923	0.0539	0.958	0.0639	0.989	0.0834
12	0.98	0.878	0.0204	0.931	0.0243	0.988	0.0318
13	0.78	0.8765	0.2171	0.9385	0.2576	0.9845	0.3348
14	0.7	0.9035	0.0946	0.947	0.1125	0.993	0.146
15	0.72	0.9675	0.1988	0.985	0.2355	0.9985	0.3055
16	0.75	0.901	0.2807	0.943	0.3314	0.987	0.4288
Conf.Int.		[0.86, 0.93]	[0.07, 0.15]	[0.90, 0.96]	[0.08, 0.18]	[0.93, 1.00]	[0.11, 0.24]

Table 2 (continued)

	G-score						
		$\delta = 0.90$		$\delta = 0.95$		$\delta = 0.99$	
Data set number	avg($\mathcal{I}(v)$)	Coverage probability	Average length of CI	Coverage probability	Average length of CI	Coverage probability	Average length of CI
G-score							
1	0.89	0.8315	0.0836	0.898	0.1019	0.948	0.1423
2	0.94	0.774	0.0442	0.812	0.0564	0.8645	0.0814
3	0.25	0.955	0.0874	0.9835	0.1034	0.998	0.1346
4	0.44	0.933	0.1011	0.9755	0.1199	0.9955	0.1562
5	0.64	0.903	0.0992	0.961	0.1175	0.99	0.1524
6	0.99	0.8305	0.0115	0.8275	0.0145	0.8805	0.0226
7	0.07	0.993	0.1297	0.9995	0.156	1	0.207
8	0.91	0.8435	0.0696	0.923	0.084	0.9765	0.1113
9	0.07	1	0.0266	1	0.0315	1	0.0409
10	0.97	0.8435	0.019	0.9095	0.0227	0.9715	0.0301
11	0.73	0.9305	0.0359	0.9635	0.0426	0.9965	0.0554
12	0.95	0.8755	0.022	0.9425	0.0262	0.986	0.0341
13	0.57	0.886	0.2764	0.948	0.329	0.99	0.4293
14	0.37	0.945	0.0897	0.9745	0.1062	0.9955	0.1379
15	0.18	0.974	0.1433	0.984	0.1694	0.999	0.2209
16	0.48	0.9005	0.2604	0.959	0.3085	0.9875	0.4004
Conf.Int.		[0.87, 0.94]	[0.05, 0.14]	[0.91, 0.97]	[0.06, 0.16]	[0.95, 1.00]	[0.08, 0.21]

Table 2 (continued)

	F_1-score						
		$\delta = 0.90$		$\delta = 0.95$		$\delta = 0.99$	
Data set number	avg($\mathcal{I}(v)$)	Coverage probability	Average length of CI	Coverage probability	Average length of CI	Coverage probability	Average length of CI
F_1-score							
1	0.96	0.8355	0.0847	0.9105	0.1057	0.9495	0.1492
2	0.98	0.78	0.0449	0.815	0.0574	0.814	0.0832
3	0.85	0.962	0.0884	0.9855	0.1048	0.998	0.1365
4	0.82	0.9455	0.1021	0.9705	0.1214	0.996	0.1581
5	0.83	0.9065	0.0997	0.957	0.1184	0.9945	0.1538
6	1.00	0.8265	0.0113	0.8225	0.0151	0.8345	0.0215
7	0.93	0.996	0.1352	0.9995	0.1639	1	0.2204
8	0.96	0.861	0.0715	0.927	0.0855	0.98	0.1147
9	0.90	1	0.0281	1	0.0334	1	0.0433
10	0.99	0.85	0.0189	0.914	0.0226	0.976	0.03
11	0.94	0.9145	0.0362	0.968	0.0429	0.9925	0.0558
12	0.98	0.869	0.0222	0.935	0.0263	0.9825	0.0342
13	0.74	0.9055	0.2866	0.946	0.3419	0.989	0.4409
14	0.75	0.942	0.0905	0.9795	0.1075	0.9965	0.1394
15	0.82	0.9725	0.1475	0.9935	0.1745	0.9995	0.2278
16	0.80	0.914	0.2659	0.9555	0.317	0.99	0.4145
Conf.Int.		[0.87, 0.94]	[0.05, 0.14]	[0.91, 0.97]	[0.06, 0.17]	[0.94, 1.00]	[0.08, 0.22]

Algorithm 3 Experimental design

```
1:  function TEST(𝓛, ℐ, δ)
2:      score ← 0
3:      for l ← 1 to2000 do
4:          (𝒟, 𝒯, 𝒮) ← randomSplit(𝓛)
5:          ℛ ← randomForest(𝒟)
6:          v ← getConfusionMatrix(ℛ, 𝒯)
7:          g ← MC-Sampler(ℐ, α^0, v, 1000)        ▷ See algorithm (1)
8:          𝒞 ← getCredibleInterval(g, δ)
9:          v_s ← getConfusionMatrix(ℛ, 𝒮)
10:         if ℐ(v_s) ∈ 𝒞 then
11:             score ← score + 1
12:         end if
13:     end for
14:     return score/2000
15: end function
```

For the case number 6 in Table 2, the coverage probability is lower than target δ. In this instance, $\mathcal{I}(v_s)$ only takes a small set of distinct values.[5] That is why we observe waves in the density estimation of $\mathcal{I}(v_s)$ in the left part of Fig. 4. In these conditions, the Bayesian method tends to overgenerate samples with $g = 1$. In the figure, we observe a high density for the value one (hatched lines). As the credible intervals $\mathcal{C}$ are taken from g, $\mathcal{C}$ tends to be too small, and therefore the coverage probabilities are lower than δ. We empirically observe that this phenomenon happens when the distribution is close to the boundary of the values the performance indicator can take.[6] In the right part of Fig. 4, we see that g follows $\mathcal{I}(v_s)$ better. That is why for case 13, the coverage probabilities are almost equal to δ in the F-Score of Table 2.

We also did a numerical comparison with the bootstrap approach. To achieve this, we reused the same experimental design described in algorithm 3 where, at line 7, algorithm 1 is replaced by algorithm 2. The experimental results are to be found in Table 3, where the *Coverage probability* and the *Average length of CI* are only computed for $\delta = 0.90$.

Compared to the previous results, the average lengths of credible intervals $\mathcal{C}$ are always smaller than those produced by the proposed approach.[7] This has already been observed in the left part of Fig. 3. The fact that the bootstrap approach tends to produce narrower credible intervals could explain why the *Coverage probability* is lower in Table 3. We can even see that the *t-based* confidence interval of the *Coverage probability* at the last line of Table 3 never contains the target confidence $\delta = 0.90$. This means that the sizes of the credible intervals are too small. By taking into account the ignorance regarding the parameters of the multinomial distribution, the Bayesian approach tends to produce distributions with higher variance.

[5] On the 2,000 loops, the function $\mathcal{I}(v_s)$ was equal to one 1,395 times.

[6] We did the experiment with the bootstrap method presented in Section 6, and we observed that bootstrap also has difficulties with these cases.

[7] For instance, in Table 2 with $\delta = 0.90$, the average length of credible intervals for the accuracy is 0.1058 when the Bayesian approach is used on the data set number 1. The corresponding average length in Table 3 is 0.0842.

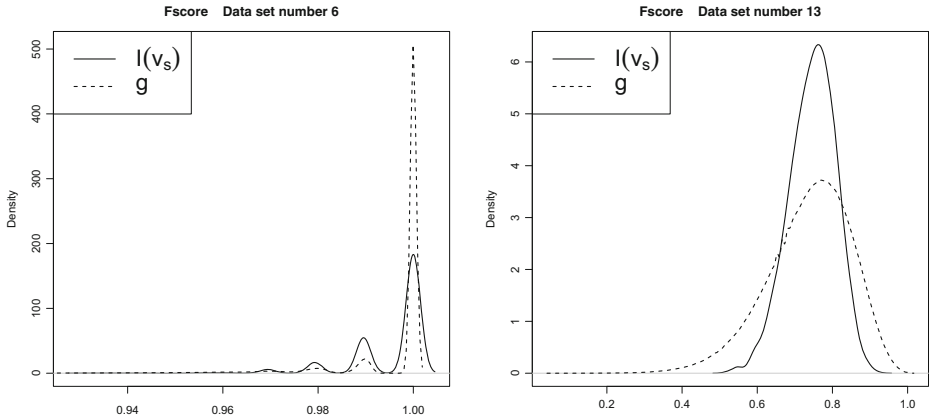

Fig. 4 Left: The distribution of $\mathcal{I}(v_s)$ and g for the F_1-score on the data set 6. Right: The distribution of $\mathcal{I}(v_s)$ and g for the F_1-score on the data set 13

7.2 Testing models sequentially

We will now evaluate the possibility of using the Bayesian interpretation of the confusion matrix in an online context.

Table 3 Experimental results with the bootstrap approach

	Accuracy		G-score		F_1-score	
	$\delta = 0.90$		$\delta = 0.90$		$\delta = 0.90$	
Data set number	Coverage probability	Average length of CI	Coverage probability	Average length of CI	Coverage probability	Average length of CI
1	0.6130	0.0842	0.7330	0.0658	0.7825	0.0687
2	0.7165	0.045	0.7115	0.0345	0.7140	0.0351
3	0.8275	0.0871	0.8650	0.0619	0.8935	0.0627
4	0.7670	0.0859	0.8390	0.0719	0.8475	0.0725
5	0.7355	0.0688	0.8005	0.0704	0.8045	0.0707
6	0.7955	0.0117	0.8285	0.0088	0.8445	0.0088
7	0.9845	0.1782	0.9790	0.0985	1	0.1019
8	0.6575	0.0577	0.7675	0.0531	0.7695	0.0535
9	1	0.0328	0.9985	0.0188	1	0.0199
10	0.6960	0.0153	0.7465	0.0144	0.7480	0.0142
11	0.7880	0.038	0.7995	0.0255	0.8105	0.0257
12	0.7355	0.0146	0.7515	0.0157	0.7675	0.0159
13	0.7405	0.1551	0.7790	0.1987	0.8055	0.2045
14	0.7710	0.0668	0.8350	0.0637	0.8425	0.0642
15	0.8905	0.1406	0.8890	0.1023	0.9185	0.1046
16	0.7835	0.1996	0.7840	0.1880	0.8125	0.1903
Conf.Int.	[0.73, 0.84]	[0.05, 0.11]	[0.78, 0.86]	[0.04, 0.1]	[0.79, 0.88]	[0.04, 0.1]

In the following experiment, the model is evaluated sequentially on a series of small testing data sets $(\mathcal{T}_1, \ldots, \mathcal{T}_l, \ldots, \mathcal{T}_L)$ and, at each step l, a confusion matrix v_l is computed from $\mathcal{T}_l$. The fact that the same model is tested sequentially gives the possibility of improving at each step l the knowledge that we have about the performance accuracy. Thanks to the Bayesian framework, at step l, we have an elegant way of injecting prior knowledge that we have accumulated during the previous steps. At each step, a credible interval will be computed and we will check over time whether it contains the true value or not. Tiny intervals will be used. This will make it possible to see if a small credible interval can contain the true value of the performance indicator. To get such small intervals, we will put the confidence level at 10%.

We don't have access to the true performance of the models. To get a good estimation of the true value of the performance indicator, we need a very extensive data set, and that is why we will synthetically generate the data from the following equation:

$$Y = \begin{cases} \vartheta_1, \ if \ \ \sin(X_1^2 + X_2^2) + \cos(X_1^2) + \cos(X_2^2) + \epsilon > 0.5 \\ \vartheta_0, \ else \end{cases} \tag{9}$$

where

$$X_1 \sim Unif(0, 3), \quad X_2 \sim Unif(0, 3) \quad \text{and} \quad \epsilon \sim N(0, 1).$$

The experimental design is described in the algorithm 4, where the MCC is used as performance indicator $\mathcal{I}$.

Algorithm 4 Testing in a sequential design

```
1:  function TEST SEQUENTIAL(I)
2:      D ← get 1,000 observations from equation (9)
3:      R ← randomForest(D)
4:      T ← get 20,000,000 observations from equation (9)
5:      v_big ← getConfusionMatrix(R, T)
6:      α ← (0, 0, 0, 0)
7:      for l ← 1 to 300 do
8:          T_l ← get 100 observations from equation (9)
9:          v_l ← getConfusionMatrix(R, T_l)
10:         g ← MC-Sampler(I, α, v_l, 1000)          ▷ See algorithm (1)
11:         C_l ← getCredibleInterval(g, 0.1)
12:         α ← α + v_l
13:     end for
14:     return [I(v_big), (C_1, ..., C_300)]
15: end function
```

The model that we will evaluate sequentially is built between lines 2 and 3. The learning data set $\mathcal{D}$ is created at line 2 with 1,000 samples generated from (9), and a random forest algorithm is used at line 3 to generate the model. This model is stored in $\mathcal{R}$. We will now try to estimate the true accuracy of the model, and for that purpose, another data set is generated at line 4 from the same (9). This data set contains twenty million samples and is used at line 5 to compute a confusion matrix v_{big}. We use twenty million samples to obtain a good estimation of the (unknown) true accuracy of $\mathcal{R}$. This confusion matrix v_{big} will be returned as output of the function at line 14. At line 6, we initialize the sequential tests by putting $\alpha = (0, 0, 0, 0)$. This stands for complete ignorance at the first loop (that is $l = 1$).

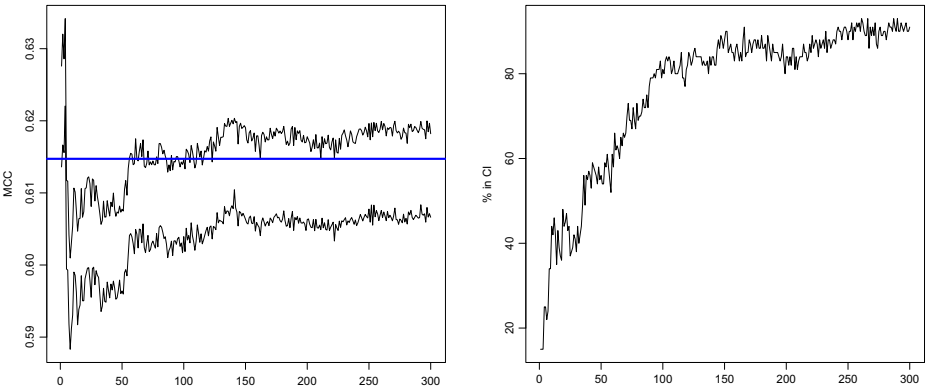

Fig. 5 Left: A run of algorithm 4. The horizontal line is the value of $\mathcal{I}\left(v_{big}\right)$. Right: After 200 runs of algorithm 4, we count the number of times that the true value is in the credible interval

The sequential tests are done between lines 7 and 13. At line 8, a small data set with only 100 observations is generated from (9). This data set is used at line 9 to compute a confusion matrix. The sampling algorithm 1 is used at line 10. To get a credible interval $\mathcal{C}_l$, at line 11, hpd is extracted from the samples in g. We want small credible intervals, which is why the confidence is put at 10%. At line 12, the *a priori* is updated with the new values for the next step. At line 14, the list with the 300 confidence intervals is returned as well as the value of the indicator function computed on the big data set.

The result of a run of algorithm 4 is available in the left part of Fig. 5. We observe in Fig. 5 that the Bayesian method provides a way to add the *a prior* knowledge from the previous loops and that, at 150 loops, the true value is always within the boundaries defined by credible intervals.

We have repeated this process 200 times and counted, for each step l, the number of times the credible interval $\mathcal{C}_l$ contained the true value. The result is available in the right part of Fig. 5. Although the confidence interval is very small, we see that after 150 loops, the true values are very often within the boundaries defined by the credible intervals (approximately 90% of the time).

8 Conclusion

In this paper, we have proposed a new way of dealing with the uncertainties a scientist can have about the performance indicators of a classifier (s)he can extract from a confusion matrix. In our work, we assume that the scientist does not have access to the learning or testing set. (S)he only has access to the confusion matrix. We have shown that the values of said matrix can be assumed to be generated from a random vector following a multinomial distribution, and by taking the Bayesian point of view, we have assumed that the unknown parameters of the multinomial distribution themselves are generated from a random vector following a Dirichlet distribution. We made a theoretical and empirical comparison between the bootstrap method and our method, based on the Bayesian framework. With α^0 used as *prior*, we showed that both methods returned samples with the same mean and that the variance of the distribution generated by bootstrap was lower. This can be explained by the fact that the bootstrap method does not assume uncertainty about the unknown parameters of

the multinomial distribution. Thanks to the Bayesian framework, we have shown that prior knowledge can easily be injected in the distributions, and that it reduces the uncertainty we can have about the distribution of the performance indicators. Experimental results show that our method can be used to construct accurate confidence intervals for the unknown performance indicator.

References

1. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240. ACM, New York (2006)
2. Efron, B.: Bootstrap methods: another look at the jackknife. In: Breakthroughs in Statistics, pp. 569–593. Springer, Berlin (1992)
3. Elkan, C.: The foundations of cost-sensitive learning. In: International Joint Conference on Artificial Intelligence, vol. 17, pp. 973–978. Lawrence Erlbaum Associates Ltd (2001)
4. Forbes, C., Evans, M., Hastings, N., Peacock, B.: Statistical distributions. Wiley, Hoboken (2011)
5. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian data analysis, vol. 2. Chapman & Hall/CRC Boca Raton, FL (2014)
6. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: Advances in Information Retrieval, pp. 345–359. Springer, Berlin (2005)
7. James, G., Witten, D., Hastie, T., Tibshirani, R.: An introduction to statistical learning, vol. 6. Springer, Berlin (2013)
8. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002). http://CRAN.R-project.org/doc/Rnews/
9. Powers, D.M.: Evaluation: from precision, recall and f-measure to roc, informedness markedness and correlation (2011)
10. Wackerly, D., Mendenhall, W., Scheaffer, R.: Mathematical statistics with applications. Cengage Learning, Boston (2008)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com