

---

# Robustness quantification: a new method for assessing the reliability of the predictions of a classifier

---

Adrián Detavernier<sup>1</sup>

Jasper De Bock<sup>1</sup>

<sup>1</sup>Foundations Lab for imprecise probabilities, Ghent University, Belgium

## ABSTRACT

Based on existing ideas in the field of imprecise probabilities, we present a new approach for assessing the reliability of the individual predictions of a generative probabilistic classifier. We call this approach robustness quantification, compare it to uncertainty quantification, and demonstrate that it continues to work well even for classifiers that are learned from small training sets that are sampled from a shifted distribution.

**Keywords.** Robustness quantification, classification, reliability, distribution shift, small data sets, imprecise probabilities.

## 1. INTRODUCTION

Let's say that you start using an AI model for a high-risk application, such as medical diagnosis or self-driving cars. At that point, if you use a classifier to automatically make predictions, it does not suffice that your model generally has a 99% accuracy: you also want to know whether the current prediction is part of that other 1% of the cases. That is, you'd really want to know how reliable the current prediction is.

Several approaches for quantifying this reliability exist, two of which we intend to compare. The most common approach is uncertainty quantification [12, 13, 19], whose aim it is to numerically quantify the amount of uncertainty associated with a prediction. We instead focus on an alternative approach, which we call robustness quantification, whose aim it is to quantify how robust a prediction is against uncertainty. One might say that we quantify how much uncertainty the model *could* cope with before changing its prediction, and this regardless of how much uncertainty is actually present. While our proposed terminology is new, the ideas behind robustness quantification have been successfully tested several times by now [6, 7, 15], always relying on techniques from imprecise probabilities [1]. Our contribution consists in conceptually introducing this approach in a general setting and comparing it with uncertainty quantification.

In particular, we focus on how uncertainty and robust-

ness quantification compare in cases where the available data is limited or when there is a distribution shift between the train and test data. Our motivation for this comparison is fueled by the fact that these problems arise frequently in practice, and can have a big impact on how a classifier performs in the real world (on unseen data) [14, 17, 18, 22]. For now, we mainly focus on the naive Bayes classifier as a test case because of its simplicity and efficiency. However, our methods can be applied to more complex classifiers as well.

From our experiments we conclude that robustness quantification keeps performing well even with limited data or in the presence of a distribution shift, while uncertainty quantification sees a decrease in performance. Moreover, our robustness metrics keep on performing well even without distribution shift and with enough data, meaning that there is no trade-off paid for the better protection that they offer.

## 2. CLASSIFICATION

We consider the problem of classification, where the goal is to predict the correct class of an instance based on its features. An example of a high-risk classification problem is predicting whether a patient has cancer (the class) based on its medical data and images (the features).

**2.1. Classifiers.** We denote the class variable by  $C$ , which can take on any value  $c$  in the finite set of classes  $\mathcal{C}$ . An instance can have one or several features  $F_i$ , with  $i \in \{1, \dots, N\}$ , where  $N$  is the number of features. Every one of these features takes values  $f_i$  in the finite set  $\mathcal{F}_i$ . The vector containing all the feature variables  $F_i$  is denoted as  $F = (F_1, \dots, F_N)$ . The particular feature values  $f_i$  of an individual instance are collected in a feature vector  $f = (f_1, \dots, f_N)$ , to which we often refer as the (set of) features of said instance. All such possible feature vectors are collected in the set  $\mathcal{F} = \mathcal{F}_1 \times \dots \times \mathcal{F}_N$ . Every instance is determined by the combination of a class  $c$  and its features  $f$ . In practice, however, we typically only know the features of an instance. The aim of a classifier is then to predict the unknown class, given those features.

Formally, this is done with a classifier: a deterministic function  $h : \mathcal{F} \rightarrow \mathcal{C}$  that maps each feature vector  $f \in \mathcal{F}$

to a class  $h(f) \in \mathcal{C}$ . Ideally, we would want a classifier to always predict the correct class. This is not possible though because instances with the same set of features may nevertheless have different classes. We then want a classifier to be correct as often as possible.

In the following sections, we dive deeper into how classifiers, and probabilistic ones in particular, predict the class of an instance, and discuss some of the problems that arise when trying to learn such classifiers from data.

**2.2. Generative probabilistic classifiers.** Rather than predict the class directly, a probabilistic classifier first predicts the (conditional) probability  $p(c|f)$  of every possible class  $c$  given the features  $f$  of the considered instance, and then predicts the class of this instance based on these probabilities. This predicted class is usually taken to be the one that has the highest probability given the features:

$$h(f) \in \arg \max_{c \in \mathcal{C}} p(c|f). \quad (1)$$

In most cases, this inclusion is simply an equality. However, it is possible, although very unlikely in practice, that there are several equiprobable classes that all have the highest probability. The set  $\arg \max_{c \in \mathcal{C}} p(c|f)$  then contains these classes, and  $h(f)$  is then arbitrarily taken to be one of them. The chosen or predicted class  $h(f)$  is called the *prediction*. If the feature vector and classifier are clear from the context, we will sometimes denote the predicted class  $h(f)$  by  $\hat{c}$  as well.

A probabilistic classifier can predict the (conditional) probabilities of the possible classes given the features in, essentially, two ways, that each correspond to a type of probabilistic classifier: discriminative or generative. Discriminative classifiers try to determine the conditional probabilities  $p(c|f)$  directly. We instead focus on generative classifiers, which first determine a joint probability distribution  $P$  for the class and features. Since  $\mathcal{C}$  and  $\mathcal{F}$  are finite in this contribution, every such distribution  $P$  is uniquely characterized by its (probability) mass function  $p : \mathcal{C} \times \mathcal{F} \rightarrow [0, 1]$ , which simply assigns a probability  $p(c, f) := P(C = c \text{ and } F = f)$  to all  $c \in \mathcal{C}$  and  $f \in \mathcal{F}$ . The conditional probabilities that are used to classify instances are then obtained by conditioning:  $p(c|f) = p(c, f)/p(f)$ , assuming that the observed features  $f$  have a non-zero (marginal) probability  $p(f) := \sum_{c \in \mathcal{C}} p(c, f) > 0$ . To emphasize the dependency of a generative classifier on the mass function  $p$ , we make this explicit in our notation by denoting it as  $h_p$ . Our reason for focussing on generative classifiers is that our ideas about robustness, and the theory of imprecise probabilities on which they are based, can be applied more naturally in this setting.

Since we maximize over the possible classes to get the prediction  $h_p(f)$  for a feature vector  $f$ , the denominator  $p(f)$  in Bayes's rule—which does not depend on the class—can be ignored in Equation (1). For generative

classifiers, Equation (1) therefore simplifies as follows:

$$h_p(f) \in \arg \max_{c \in \mathcal{C}} p(c, f). \quad (2)$$

If  $p(f) = 0$ , then  $p(c, f) = 0$  for all  $c$  as well, in which case Equation (2) tells us that  $h_p(f)$  is an arbitrary class in  $\mathcal{C}$ ; this case is typically avoided in practice though.

**2.3. Learning a classifier.** If we want to use a (generative) classifier, we first need to somehow learn its joint distribution  $P$ . This is typically done with a labeled data set, where the class of each instance is known. We will call this data set the training set, and denote it by  $D_{\text{train}}$ . In practice, it is typically assumed that the training set has an underlying distribution that generated it: the training distribution  $P_{\text{train}}$ . Once a classifier has been learned, we of course want to put it to use on unseen data to see how well it performs. To that end, we consider a second data set  $D_{\text{test}}$  that we call the test set. Here too, it is typically assumed that this data set has an underlying distribution, called the test distribution, which we denote by  $P_{\text{test}}$ . In a perfect scenario then, the joint distribution  $P$  of our classifier is equal to  $P_{\text{test}}$ , yielding a classifier that issues the correct prediction as often as possible, at least on average for large test sets  $D_{\text{test}}$ . However, this almost never occurs because of, essentially, two reasons. One reason is that  $P_{\text{test}}$  may not be equal to  $P_{\text{train}}$ , which is what we will come to next, but let us first look at the case where they are equal.

If  $P_{\text{test}} = P_{\text{train}}$ , then the task of learning a generative classifier basically boils down to trying to make sure that  $P$  approximates  $P_{\text{train}}$  sufficiently well, the perfect scenario being that  $P = P_{\text{train}} = P_{\text{test}}$ . However, this perfect scenario is only possible in the idealized situation where  $D_{\text{train}}$  is infinitely large. For (very) small training sets  $D_{\text{train}}$ , in fact, it often happens that  $D_{\text{train}}$  is not at all representative for  $P_{\text{train}}$ , making it impossible for  $P$  to approximate  $P_{\text{train}}$  with any reasonable accuracy. A possible solution to this problem consists in collecting more data. Unfortunately, this not always possible.

Regardless of whether  $D_{\text{train}}$  is large enough to be able to accurately learn  $P_{\text{train}}$ , there is also the additional problem that  $P_{\text{train}}$  may not be equal to  $P_{\text{test}}$ , a phenomenon known as *distribution shift*. Consider for example a situation where a classifier is trained based on medical data from one hospital, and then used in another hospital [24], or where it is trained on colored images but then used on black-and-white images. In such cases, even if  $P = P_{\text{train}}$ , the performance of the corresponding classifier  $h_p$  is bound to suffer. This is a very common problem, that is known to have a big impact on the performance of classifiers [14, 17]. Distribution shift is often ignored though, by (tacitly) assuming that  $P_{\text{train}}$  and  $P_{\text{test}}$  are equal.

Regardless of how well  $P$  approximates  $P_{\text{test}}$ , the performance of a classifier is often measured in terms of *accuracy*: the number of correctly predicted instances

divided by the total number of predictions. This metric is, in most cases, good enough to evaluate a classifier, since the goal is often to be correct as much as possible. However, in cases where a bad prediction could have huge consequences, we are no longer only interested in the overall performance of the classifier. In such cases, it is equally important to assess the reliability of each individual prediction, allowing us to flag those that might be unreliable. This brings us to reliability quantification.

### 3. RELIABILITY QUANTIFICATION

In high-risk applications, it is of crucial importance to know for each separate prediction whether it is correct or not, because a wrong prediction could have severe consequences. Although it is of course infeasible to construct a classifier that is always correct, we could try to construct a classifier that is correct in most cases, and can single out the other, more difficult cases. This is exactly what we want to achieve with *reliability quantification*. That is, we want to quantify the reliability of each individual prediction. Any numeric value that aims to quantify (some aspect of) the reliability of a prediction, we call a *reliability metric*. To stress that these reliability metrics are associated with individual predictions, instead of the overall performance of a classifier, we say that they are *instance-based*. We restrict ourselves to two methods for quantifying instance-based reliability, each of which focuses on a different aspect of reliability: *uncertainty quantification* and *robustness quantification*.

**3.1. Uncertainty.** Uncertainty quantification, as the terminology suggests, aims to quantify the uncertainty associated with individual predictions, the idea being that uncertainty negatively influences reliability. To better understand some of the challenges of this approach, it is helpful to distinguish two different types of uncertainty that are often considered in this context [13]: *aleatoric* and *epistemic* uncertainty.

Aleatoric uncertainty has to do with the intrinsic variability of the test data, as captured by  $P_{\text{test}}$ . For an instance with features  $f$ , it is the uncertainty that corresponds to the conditional probabilities  $p_{\text{test}}(c|f)$ : even if we know  $P_{\text{test}}$  perfectly, then still, several classes  $c$  remain possible, each with their own probability of occurrence  $p_{\text{test}}(c|f)$ . Since it is intrinsic to the process, aleatoric uncertainty is irreducible, meaning that it will always remain no matter how hard we try; it is the reason why in practice no classifier can be expected to always yield a correct prediction.

Epistemic uncertainty, on the other hand, has to do with the fact that we typically do not know  $P_{\text{test}}$ , in the sense that the joint distribution  $P$  of a classifier will typically not be equal to  $P_{\text{test}}$ . One aspect of this epistemic uncertainty has to do with the fact that  $P \neq P_{\text{train}}$ , due to modelling choices or the limited availability of training data; this could be addressed by collecting more data,

and is therefore considered reducible. Another aspect of epistemic uncertainty has to do with the presence of distribution shift ( $P_{\text{train}} \neq P_{\text{test}}$ ), which can only be reduced if we have access to labelled data from  $P_{\text{test}}$ .

The aim of uncertainty quantification, then, is to numerically quantify these two types of uncertainty with uncertainty metrics. This is extremely challenging though. On the one hand, for epistemic uncertainty, there is typically no way of knowing to which extent  $D_{\text{train}}$  is representative for  $P_{\text{train}}$ , nor how much  $P_{\text{test}}$  differs from  $P_{\text{train}}$ , making it very difficult to assess to which degree  $P$  differs from  $P_{\text{test}}$ . On the other hand, uncertainty metrics that aim to quantify aleatoric uncertainty would ideally be based on the conditional probabilities  $p_{\text{test}}(c|f)$ . In practice, however,  $P_{\text{test}}$  is not available, and one then has to do with the estimated conditional probabilities  $p(c|f)$  instead, which may differ considerably from the real ones  $p_{\text{test}}(c|f)$  due to the presence of epistemic uncertainty.

Nevertheless, many *uncertainty metrics* have been developed, some of which we discuss in more detail in Section 4. Some of these metrics try to quantify aleatoric and epistemic uncertainty separately, while others try to quantify the combined effect of both, to which they refer as total uncertainty. However, one might question to which extent it is possible to make these distinctions while quantifying uncertainty though, since both notions are always intertwined: as we have just explained, aleatoric uncertainty cannot be reliably quantified without quantifying epistemic uncertainty, and epistemic uncertainty itself is very hard to quantify. It is therefore to be expected that these methods will perform significantly worse in the presence of epistemic uncertainty, as we will also demonstrate in our experiments.

**3.2. Robustness.** Motivated by these difficulties, we here put forward a different approach, called robustness quantification. The main idea is to approach reliability from a different angle: instead of quantifying its uncertainty, which is notoriously hard to do, we instead quantify the robustness of a prediction. That is, we quantify how much epistemic uncertainty the classifier could handle without changing its prediction, thereby completely avoiding the difficult task of quantifying how much epistemic uncertainty there actually is. A prediction is then deemed reliable if it is robust, meaning that it would remain true even in the face of severe epistemic uncertainty. We formalize this concept in Section 5 for general generative classifiers, and further develop it in Section 6 for the particular case of Naive Bayes classifiers.

The main feature that sets our approach apart from other approaches that consider the robustness of a classifier, such as adversarial robustness [2, 4] or robustness against distribution shift [21], is its instance-based character: we consider the robustness of individual predictions, whereas these other approaches consider the

robustness of the overall performance of a classifier. For adversarial robustness [2, 4], another difference is that we consider robustness against epistemic uncertainty (resulting from, for example, distribution shift), instead of robustness against manipulations of the (in that case typically continuous) features.

#### 4. UNCERTAINTY QUANTIFICATION

Before we delve into the topic of robustness quantification, and robustness metrics in particular, let us first take a brief look at a number of uncertainty metrics. The first two metrics single out a specific aspect of the uncertainty about the class given the features: the probability of the predicted class and the (Shannon) entropy, respectively. The last three metrics are also based on entropy, but combine this with ensemble techniques to better estimate the uncertainty [20]. Each of these five uncertainty metrics can be applied to arbitrary generative classifiers. For a more extensive overview, which also covers metrics for discriminative classifiers, or for specific model architectures, we refer the interested reader to the work of Hüllermeier and Waegeman [13].

*Maximum probability.* A first straightforward uncertainty metric, denoted by  $u_m$ , is one minus the conditional probability of the predicted class  $\hat{c}$ , or equivalently, one minus the maximum probability over all classes:

$$u_m(f) = 1 - \max_{c \in \mathcal{C}} p(c|f) = 1 - p(\hat{c}|f). \quad (3)$$

In an idealized situation where there is no epistemic uncertainty (so  $P = P_{\text{test}}$ ), this is the probability of making an incorrect prediction, and hence a perfect metric for quantifying aleatoric uncertainty. In general, in the presence of epistemic uncertainty, this might then be taken to represent a combination of both types.

*Entropy.* A second metric for quantifying uncertainty, which is quite popular and has its roots in information theory, is *entropy*. In particular, we consider the entropy of the conditional probability mass function for the class given the features  $f$ :

$$u_H(f) = - \sum_{c \in \mathcal{C}} p(c|f) \log_2 p(c|f). \quad (4)$$

Entropy mostly tells us something about the shape of the conditional probability mass function: the more uniform it is, the higher the entropy. Since the uniform case corresponds to randomly guessing the correct class, it makes sense to associate high values for  $u_H(f)$  with there being more uncertainty; in particular, it is typically taken to be a metric that aims to capture the total uncertainty [20].

The last three uncertainty metrics combine entropy with ensemble techniques to quantify aleatoric, total and epistemic uncertainty, respectively. Instead of training the classifier once on the training set  $D_{\text{train}}$ , we now construct several bootstrap samples (obtained by sampling

from  $D_{\text{train}}$  set with replacement to obtain a data set of the same size as  $D_{\text{train}}$ ) and each of these samples train a new classifier. The number of bootstrap samples, and thus the number of classifiers we train, is denoted by  $M$ . We denote the predicted joint probability mass functions of these different classifiers by  $p_i$ , with  $i \in \{1, \dots, M\}$ . In our experiments in Section 7, we use  $M = 10$ .

*Aleatoric uncertainty.* The *aleatoric uncertainty* is then estimated as follows:

$$u_a(f) = -\frac{1}{M} \sum_{i=1}^M \sum_{c \in \mathcal{C}} p_i(c|f) \log_2 p_i(c|f). \quad (5)$$

This is the average entropy of the predicted conditional probability distributions over all the classifiers in the ensemble. The reason this is typically associated with aleatoric uncertainty, is because averaging over the ensembles is believed to reduce the influence of the epistemic uncertainty as much as possible.

*Total uncertainty.* The total uncertainty, on the other hand, is then estimated as the entropy of the average predicted conditional probability distribution, given by

$$u_t(f) = - \sum_{c \in \mathcal{C}} p_{\text{av}}(c|f) \log_2 p_{\text{av}}(c|f), \quad (6)$$

with  $p_{\text{av}}(c|f) := 1/M \sum_{i=1}^M p_i(c|f)$ . So we see that this uncertainty metric is similar to  $u_H$ , but with  $p$  replaced by the average of the different  $p_i$ .

*Epistemic uncertainty.* The epistemic uncertainty, finally, is taken to be the difference of the previous two metrics:

$$u_e(f) = u_a(f) - u_t(f). \quad (7)$$

Underlying this metric is an assumption that the total uncertainty is the sum of the aleatoric and epistemic uncertainty. This assumption seems questionable though, given that both types of uncertainty are intertwined.

#### 5. ROBUSTNESS QUANTIFICATION

To quantify the robustness of a prediction of a classifier, we need to somehow assess how much epistemic uncertainty this classifier can handle before that particular prediction will change. That is, how different  $P_{\text{test}}$  can be from  $P$  while still providing the same prediction. Inspired by ideas from the imprecise probabilities literature [6, 7, 15], we will do this by artificially perturbing (the mass function  $p$  associated with) the joint distribution  $P$ , defining a robustness metric as the minimal perturbation for which the prediction is no longer robust.

**5.1. Robustness w.r.t. a perturbation.** We define a perturbation of a mass function  $p$  as a set of mass functions that contains  $p$ .

**Definition 5.1.** Consider a mass function  $p$  on  $\mathcal{C} \times \mathcal{F}$ . Let  $\mathcal{P}$  be a compact set of mass functions on  $\mathcal{C} \times \mathcal{F}$ , with  $p \in \mathcal{P}$ . Then we call  $\mathcal{P}$  a *perturbation* of  $p$ .



This definition is quite general, as it allows for a wide range of perturbation types. Usually, however, a perturbation  $\mathcal{P}$  of  $p$  will be a ‘neighborhood’ around  $p$ , consisting of mass functions of a particular type whose distance from  $p$  is in some sense bounded.

There are several ways in which such a perturbation could be obtained. The most straightforward one, which we will adopt here, is to directly perturb  $p$  itself. However, we could also perturb the data from which the model is learned and, in this way, indirectly perturb  $p$  during the learning process; we leave this for future work.

Since a perturbation  $\mathcal{P}$  is a set of mass functions, we can associate with each such mass function  $p' \in \mathcal{P}$  a generative classifier  $h_{p'}$ —or multiple ones, if there is no unique maximiser in Equation (1). If the prediction  $h_{p'}(f)$  of all those classifiers  $h_{p'}$  is the same as the class  $\hat{c} = h_p(f)$  that is predicted by  $h_p$ , we say that the prediction of  $h_p$  is *robust* w.r.t. the perturbation  $\mathcal{P}$ .

**Definition 5.2.** Let  $h_p$  be a generative classifier corresponding to a mass function  $p$ . Let  $\mathcal{P}$  a perturbation of  $p$  and let  $\hat{c}$  be the prediction according to  $h_p$  for the set of features  $f$ . Then  $\hat{c}$  is *robust* w.r.t. the perturbation  $\mathcal{P}$  if  $\arg \max_{c \in \mathcal{C}} p'(c, f) = \{\hat{c}\}$  for all  $p' \in \mathcal{P}$ .

Since  $\mathcal{P}$  contains  $p$ , this definition requires in particular that  $\arg \max_{c \in \mathcal{C}} p(c, f) = \{\hat{c}\}$ , meaning that if the class  $\hat{c}$  predicted by the original classifier  $h_p$  does not uniquely have the highest probability, then  $\hat{c}$  can never be robust.

To check if a prediction is robust w.r.t. a perturbation  $\mathcal{P}$ , we do not need to explicitly check for each individual  $p' \in \mathcal{P}$  whether it uniquely predicts  $\hat{c}$ ; fortunately, we can instead reformulate robustness as a maximization problem. This result—and our proof—is entirely analogous to [7, Theorem 1], where a similar reformulation was presented for the case without observed features.

**Theorem 5.1.** Let  $h_p$  be a generative classifier corresponding to a mass function  $p$ . Let  $\mathcal{P}$  be a perturbation of  $p$  and let  $\hat{c}$  be the prediction according to  $h_p$  for the set of features  $f$ . Then  $\hat{c}$  is robust w.r.t. the perturbation  $\mathcal{P}$  if and only if

$$\min_{p' \in \mathcal{P}} p'(\hat{c}, f) > 0 \quad \text{and} \quad \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \max_{p' \in \mathcal{P}} \frac{p'(c, f)}{p'(\hat{c}, f)} < 1, \quad (8)$$

where the first inequality should be checked first because if it fails, the fraction in the second inequality is undefined.

*Proof.* By definition,  $\hat{c}$  is robust w.r.t.  $\mathcal{P}$  if and only if  $\arg \max_{c \in \mathcal{C}} p'(c, f) = \{\hat{c}\}$  for all  $p' \in \mathcal{P}$ . This is clearly the case if and only if

$$\forall p' \in \mathcal{P}, \forall c \in \mathcal{C} \setminus \{\hat{c}\}: p'(c, f) < p'(\hat{c}, f). \quad (9)$$

Since  $p'(c, f) \geq 0$ , this can only be true if  $p'(\hat{c}, f) > 0$  for each  $p' \in \mathcal{P}$ , or equivalently, due to the compactness of

$\mathcal{P}$ , if the first inequality of Equation (8) holds. Assuming that it holds, Equation (9) can now be rewritten as

$$\max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \max_{p' \in \mathcal{P}} \frac{p'(c, f)}{p'(\hat{c}, f)} < 1, \quad (10)$$

where the compactness of  $\mathcal{P}$  ensures that the maximum is well-defined.  $\square$

From the point of view of imprecise probabilities [1], our notion of robustness is closely related to credal—or imprecise—classification [5]. Starting from a set of probabilities  $\mathcal{P}$ —or a perturbation, in our language—this approach to classification provides a set-valued prediction  $\cup_{p' \in \mathcal{P}} \arg \max_{c \in \mathcal{C}} p'(c, f)$  that gathers the predictions  $h_{p'}(f)$  of every possible classifier  $h_{p'}$  that corresponds to a joint mass function  $p' \in \mathcal{P}$ . Robustness then corresponds to the specific situation where this (possibly) set-valued prediction consists of only one class, which is then necessarily equal to  $\hat{c}$ .

**5.2. Robustness metrics.** To be able to numerically quantify the robustness of a prediction as the minimal perturbation for which it is no longer robust, we need to somehow express what the size of perturbation is. To this end, we will make use of parametrized perturbations.

**Definition 5.3.** Consider a probability mass function  $p$  on  $\mathcal{C} \times \mathcal{F}$  and, for all  $\varepsilon \in [0, 1]$ , a perturbation  $\mathcal{P}_\varepsilon$  of  $p$ . Then the family  $\mathcal{P} := (\mathcal{P}_\varepsilon)_{\varepsilon \in [0, 1]}$  is called a *parametrized perturbation* of  $p$  if it satisfies the following conditions:

- if  $\varepsilon = 0$ , then  $\mathcal{P}_0 = \{p\}$ ;
- if  $\varepsilon_1 < \varepsilon_2$ , then  $\mathcal{P}_{\varepsilon_1} \subset \mathcal{P}_{\varepsilon_2}$ .

So the bigger the parameter  $\varepsilon$ , the bigger the perturbation. The idea is now to increase  $\varepsilon$  until the prediction is no longer robust w.r.t.  $\mathcal{P}_\varepsilon$ , and to use the value of  $\varepsilon$  for which this happens as a robustness metric.

**Definition 5.4.** Let  $h_p$  be a generative classifier corresponding to a mass function  $p$ , and let  $\hat{c}$  be the prediction according to  $h_p$  for the set of features  $f$ . Let  $\mathcal{P}_\varepsilon$  be a parametrized perturbation of  $p$ . Then the *robustness metric*  $\varepsilon_{\mathcal{P}_\varepsilon}(f)$ —if it exists—is the smallest value of  $\varepsilon$  for which  $\hat{c}$  is no longer robust w.r.t.  $\mathcal{P}_\varepsilon$ .

An important advantage of this approach, compared to both (epistemic) uncertainty quantification and credal classification, is that it avoids the problem of determining which perturbation  $\mathcal{P}_\varepsilon$ —or which  $\varepsilon \in [0, 1]$ —is ‘correct’.

That said, there are of course many ways to construct a parametrized perturbation  $\mathcal{P}_\varepsilon$ , and hence many robustness metrics that can be obtained in this way. The ones we will focus on are based on so-called  $\varepsilon$ -contamination [3].

**Definition 5.5.** Consider a probability mass function  $p$  on a set  $\mathcal{X}$ , and let  $\Sigma_{\mathcal{X}}$  be the set of all possible probability mass functions on  $\mathcal{X}$ . Then for any  $\varepsilon \in [0, 1]$ , we define

the  $\varepsilon$ -contamination of  $p$  as the family of all mass functions that are convex mixtures, with mixture coefficient  $\varepsilon$ , of  $p$  and an arbitrary element of  $\Sigma_X$ :

$$\mathcal{M}_{p,\varepsilon} = \{(1-\varepsilon)p + \varepsilon p^* : p^* \in \Sigma_X\}. \quad (11)$$

In particular, a first type of family of perturbations that we will consider, is the one obtained by directly applying  $\varepsilon$ -contamination to the learned probability mass function  $p$ . We call this family the *global* parametrized perturbation of  $p$  and denote it by  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$ . For all  $\varepsilon \in [0, 1]$ , the corresponding perturbation of  $p$  is given by

$$\mathcal{P}_{p,\varepsilon}^{\text{glob}} := \mathcal{M}_{p,\varepsilon}. \quad (12)$$

We call the robustness metric that corresponds to  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$  the *global robustness metric* and, for notational convenience, denote it by  $\varepsilon_{\text{glob}} := \varepsilon_{\mathcal{P}_{p,\varepsilon}^{\text{glob}}}$ . Our next result provides a closed-form expression.

**Theorem 5.2.** *Consider a set of features  $f$  and let  $\hat{c}$  be the prediction according to a generative classifier  $h_p$  with joint probability mass function  $p$ . Then*

$$\varepsilon_{\text{glob}}(f) = \frac{p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f)}{1 + p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f)}. \quad (13)$$

*Proof.* If  $p(\hat{c}, f) = 0$ , it follows from Equation (2) that  $p(c, f) = 0$  for all  $c \in \mathcal{C}$ , which implies that  $\hat{c}$  is not robust for  $\mathcal{P}_{p,0}^{\text{glob}} = \{p\}$  because  $\arg \max_{c \in \mathcal{C}} p(c, f) = \mathcal{C}$  is not a singleton. So in this case, we should have that  $\varepsilon_{\text{glob}}(f) = 0$ , which indeed corresponds to Equation (13).

This leaves us with the case  $p(\hat{c}, f) > 0$ . To show that (13) is correct in that case too, we fix any  $\varepsilon \in [0, 1]$ . Since  $p(\hat{c}, f) > 0$ , it follows from Equation (12) and Definition 5.5 that  $p'(\hat{c}, f) > 0$  for all  $p' \in \mathcal{P}_{p,\varepsilon}^{\text{glob}}$ . Theorem 5.1, Equation (12) and Definition 5.5 therefore imply that  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$  if and only if

$$\max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \max_{p^* \in \Sigma_{\mathcal{C} \times \mathcal{F}}} \frac{(1-\varepsilon)p(c, f) + \varepsilon p^*(c, f)}{(1-\varepsilon)p(\hat{c}, f) + \varepsilon p^*(\hat{c}, f)} \geq 1, \quad (14)$$

where  $\Sigma_{\mathcal{C} \times \mathcal{F}}$  is the set of all mass functions on  $\mathcal{C} \times \mathcal{F}$ . Since the inner maximum is clearly attained by the unique distribution  $p^* \in \Sigma_{\mathcal{C} \times \mathcal{F}}$  that assigns probability one to  $(c, f)$ —and hence zero probability to  $(\hat{c}, f)$ —the left-hand side of Equation (14) simplifies to

$$\max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \frac{(1-\varepsilon)p(c, f) + \varepsilon}{(1-\varepsilon)p(\hat{c}, f)} = \frac{(1-\varepsilon) \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f) + \varepsilon}{(1-\varepsilon)p(\hat{c}, f)} \quad (15)$$

Rearranging the terms, it therefore follows that Equation (14) holds—or equivalently, that  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$ —if and only if

$$\frac{p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f)}{1 + p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f)} \leq \varepsilon. \quad (16)$$

Furthermore, since we know from Equation (2) that  $0 \leq p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f) \leq 1$ , the left-hand side of this inequality is at most  $1/2$ . The smallest value of  $\varepsilon \in [0, 1]$ —and hence also of  $\varepsilon \in [0, 1]$ —for which  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$  is therefore equal to this left-hand side. This is exactly the value given by Equation (13).  $\square$

It follows from this result that our global robustness metric also has a alternative, simple and perhaps more intuitive interpretation that does not make use of perturbations: as can be seen from Equation (13),  $\varepsilon_{\text{glob}}$  turns out to be a monotone transformation of the difference  $p(\hat{c}, f) - \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f)$  between the joint probabilities of the most likely and second most likely classes.

## 6. ROBUSTNESS QUANTIFICATION FOR THE NAIVE BAYES CLASSIFIER

An important advantage of the parametrized perturbation  $\mathcal{P}_{p,\varepsilon}^{\text{glob}}$ , and the corresponding robustness metric  $\varepsilon_{\text{glob}}$ , is that they can be applied to any generative classifier. For specific types of classifiers, however, we can also consider more specific types of perturbations that are tailor-made, as we are about to illustrate for the simple case of the Naive Bayes Classifier [9].

**6.1. Naive Bayes Classifier.** The central assumption on which the Naive Bayes Classifier (NBC) is based is that the features are conditionally independent given the class. For all classes  $c \in \mathcal{C}$  and features  $f \in \mathcal{F}$ , this implies that the joint probability mass function can be factorized as

$$p(c, f) = p(c) \prod_{i=1}^N p(f_i | c), \quad (17)$$

where  $p(c) := \sum_{f \in \mathcal{F}} p(c, f)$  is the marginal probability of  $c$  and  $p(f_i | c)$  is the conditional probability of the feature  $f_i$  given  $c$ . We collect all probability mass functions on  $\mathcal{C} \times \mathcal{F}$  that satisfy Equation (17) in the set  $\Sigma_{\mathcal{C} \times \mathcal{F}}^{\text{NBC}}$ . Every mass function  $p$  in  $\Sigma_{\mathcal{C} \times \mathcal{F}}^{\text{NBC}}$  is completely determined by a local mass function  $p_C$  on  $\mathcal{C}$ , defined by  $p_C(c) := p(c)$  for all  $c \in \mathcal{C}$ , and, for all  $c \in \mathcal{C}$ , a local mass function  $p_{F_i|c}$  on  $\mathcal{F}_i$ , defined by  $p_{F_i|c}(f_i) := p(f_i | c)$  for all  $f_i \in \mathcal{F}_i$ .

The main advantage of assuming independence of the features given the class is that we do not need to learn the joint probability mass function  $p$  as a whole—which is often high-dimensional—but can focus on learning the—typically low-dimensional—local probability mass functions  $p_C$  and  $p_{F_i|c}$  instead. In the experiments in Section 7, the probabilities that make up these local mass functions are obtained as follows:

$$p(c) = \frac{n(c) + \alpha}{n + \alpha|\mathcal{C}|} \quad \text{and} \quad p(f_i | c) = \frac{n(c, f_i) + \alpha}{n(c) + \alpha|\mathcal{F}_i|}, \quad (18)$$

where  $n$  is the total number of training instances,  $n(c)$  is the number of instances with class  $c$ , and  $n(c, f_i)$  is the

number of instances with class  $c$  and feature value  $f_i$ . These expressions correspond to the posterior predictive distributions of the Dirichlet-multinomial model [11]. For  $\alpha = 0$ , the obtained probabilities are simply the observed relative frequencies of the different classes, and of the features given each of the classes. The addition of a small additive smoothing parameter  $\alpha > 0$  avoids that the denominator becomes zero—if not all combinations of classes and features are present in the data—and protects against overfitting. The exact value of  $\alpha$  is determined by optimizing it, using 5-fold cross validation on the training set.

The downside that comes with this advantage, on the other hand, is that the assumption that the features are conditionally independent given the class is often unrealistic—or ‘naive’—in practice. Enforcing this assumption anyway will then make it impossible for  $P$  to closely resemble  $P_{\text{train}}$ . Nevertheless, and perhaps surprisingly, the performance of NBCs is often competitive with other, more complicated types of classifiers [8, 10].

**6.2. Two types of perturbations.** For the particular case of an NBC with mass function  $p$ , we now consider two types of perturbations and their corresponding robustness metrics.

*Global perturbations.* The first family of perturbations is the global parametrized perturbation  $\mathcal{P}_{p,\bullet}^{\text{glob}}$  of Section 5.2. The expression for the corresponding robustness metric  $\varepsilon_{\text{glob}}$  does not simplify further for the specific case of an NBC, and is still given by Equation (13), in which we can easily evaluate  $p(\hat{c}, f)$  and  $p(c, f)$  with Equation (17).

*Local perturbations.* The second family of perturbations takes inspiration from the set of probabilities that defines a Naive Credal Classifier [23]—an imprecise-probabilistic generalization of an NBC. It is tailor-made for NBCs, as it makes use of the fact that  $p$  factorizes as in Equation (17). In particular, for each  $\varepsilon \in [0, 1]$ , we let

$$\mathcal{P}_{p,\varepsilon}^{\text{loc}} := \left\{ p' \in \Sigma_{\mathcal{C} \times \mathcal{F}}^{\text{NBC}} : p'_C \in \mathcal{M}_{p_C, \varepsilon}, p'_{F_i|c} \in \mathcal{M}_{p_{F_i|c}, \varepsilon} \right\} \quad (19)$$

be the perturbation of  $p$  obtained by separately perturbing each of local mass functions with  $\varepsilon$ -contamination.<sup>1</sup> We call the resulting family  $\mathcal{P}_{p,\bullet}^{\text{loc}}$  the *local parametrized perturbation* of  $p$  and call the corresponding robustness metric the *local robustness metric*. For notational convenience, denote it by  $\varepsilon_{\text{loc}} := \varepsilon_{\mathcal{P}_{p,\bullet}^{\text{loc}}}$ . Our next result gives a convenient characterization.

**Theorem 6.1.** *Consider a set of features  $f$  and let  $\hat{c}$  be the prediction according to an NBC  $h_p$  whose joint probability mass function  $p$  factorizes according to Equation (17).*

<sup>1</sup>This is not the only possibility. We could for example also perturb each of the local mass functions of the NBC by a different amount  $\varepsilon_i$  and then let  $\varepsilon$  be the sum of these different amounts.

Then  $\phi : [0, 1) \rightarrow \mathbb{R}_{\geq 0}$ , defined for all  $\varepsilon \in [0, 1)$  by

$$\phi(\varepsilon) := \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \left( p(c) + \frac{\varepsilon}{1 - \varepsilon} \right) \prod_{i=1}^N \left( p(f_i|c) + \frac{\varepsilon}{1 - \varepsilon} \right), \quad (20)$$

is a strictly increasing function and  $\varepsilon_{\text{loc}}(f)$  is the unique value of  $\varepsilon$  for which  $\phi(\varepsilon) = p(\hat{c}, f)$ .

*Proof.* Since all the factors inside the maximum are nonnegative and strictly increasing in  $\varepsilon$ , the same is clearly true for  $\phi(\varepsilon)$  itself. So it suffices to show that  $\phi(\varepsilon_{\text{loc}}(f)) = p(\hat{c}, f)$ .

If  $p(\hat{c}, f) = 0$ , it follows from Equation (2) that  $p(c, f) = 0$  for all  $c \in \mathcal{C}$ , which implies that  $\hat{c}$  is not robust for  $\mathcal{P}_{p,0}^{\text{loc}} = \{p\}$  because  $\arg \max_{c \in \mathcal{C}} p(c, f) = \mathcal{C}$  is not a singleton. So  $\varepsilon_{\text{loc}}(f) = 0$  and therefore

$$\phi(\varepsilon_{\text{loc}}(f)) = \phi(0) = \max_{c \in \mathcal{C} \setminus \{\hat{c}\}} p(c, f) = 0 = p(\hat{c}, f), \quad (21)$$

using Equation (17) for the second equality.

This leaves us with the case  $p(\hat{c}, f) > 0$ . To show that  $\phi(\varepsilon_{\text{loc}}(f)) = p(\hat{c}, f)$  in that case too, we fix any  $\varepsilon \in [0, 1)$ . Since  $0 < p(\hat{c}, f) = p(\hat{c}) \prod_{i=1}^N p(f_i|\hat{c})$ , it follows from Equation (19) and Definition 5.5 that  $p'(\hat{c}, f) > 0$  for all  $p' \in \mathcal{P}_{p,\varepsilon}^{\text{loc}}$ . It therefore follows from Theorem 5.1 that  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{loc}}$  if and only if

$$\max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \max_{p' \in \mathcal{P}_{p,\varepsilon}^{\text{loc}}} \frac{p'(c, f)}{p'(\hat{c}, f)} \geq 1. \quad (22)$$

Ignoring the maximum over the classes for now, rewriting the innermost maximum with Equation (19) yields

$$\max_{p'_C \in \mathcal{M}_{p_C, \varepsilon}} \frac{p'_C(c)}{p'_C(\hat{c})} \prod_{i=1}^N \frac{\max_{p'_{F_i|c} \in \mathcal{M}_{p_{F_i|c}, \varepsilon}} p'_{F_i|c}(f_i)}{\min_{p'_{F_i|\hat{c}} \in \mathcal{M}_{p_{F_i|\hat{c}}, \varepsilon}} p'_{F_i|\hat{c}}(f_i)}. \quad (23)$$

For the first maximum in this expression, we find that

$$\max_{p'_C \in \mathcal{M}_{p_C, \varepsilon}} \frac{p'_C(c)}{p'_C(\hat{c})} = \max_{p^* \in \Sigma_{\mathcal{C}}} \frac{(1 - \varepsilon)p_C(c) + \varepsilon p^*(c)}{(1 - \varepsilon)p_C(\hat{c}) + \varepsilon p^*(\hat{c})} \quad (24)$$

$$= \frac{(1 - \varepsilon)p(c) + \varepsilon}{(1 - \varepsilon)p(\hat{c})}, \quad (25)$$

where the maximum was attained by the unique mass function  $p^* \in \Sigma_{\mathcal{C}}$  that assigns probability one to  $c$ —and hence zero probability to  $\hat{c}$ .

Using a similar line of reasoning, we find that the maxima in the product are equal to  $(1 - \varepsilon)p(f_i|c) + \varepsilon$ , whereas the minima are equal to  $(1 - \varepsilon)p(f_i|\hat{c})$ .

So we see that Equation (22) holds if and only if

$$\max_{c \in \mathcal{C} \setminus \{\hat{c}\}} \frac{((1 - \varepsilon)p(c) + \varepsilon) \prod_{i=1}^N ((1 - \varepsilon)p(f_i|c) + \varepsilon)}{(1 - \varepsilon)p(\hat{c}) \prod_{i=1}^N (1 - \varepsilon)p(f_i|\hat{c})} \geq 1, \quad (26)$$



or equivalently,  $\phi(\varepsilon) \geq p(\hat{c}) \prod_{i=1}^N p(f_i|\hat{c}) = p(\hat{c}, f)$ , using Equation (17) for the last equality.

So, we see that for  $\varepsilon \in [0, 1]$ ,  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{loc}}$  if and only if  $\phi(\varepsilon) \geq p(\hat{c}, f)$ .

On the other hand, we already know that  $\phi(\varepsilon)$  is strictly increasing in  $\varepsilon$  and, since  $\varepsilon/(1-\varepsilon) \geq 1$  for  $\varepsilon \geq 1/2$ , we also know that  $\phi(1/2) \geq 1 \geq p(\hat{c}, f)$  for  $\varepsilon \geq 1/2$ .

It therefore follows that there is a smallest  $\varepsilon \in [0, 1]$  for which  $\hat{c}$  is not robust w.r.t.  $\mathcal{P}_{p,\varepsilon}^{\text{loc}}$ , and that this smallest value—which is by definition equal to  $\varepsilon_{\text{loc}}(f)$ —is indeed the unique value of  $\varepsilon$  for which  $\phi(\varepsilon) = p(\hat{c}, f)$ .  $\square$

Due to this result,  $\varepsilon_{\text{loc}}(f)$  can easily be evaluated in practice: it suffices to find the unique value of  $\varepsilon \in [0, 1]$  for which the strictly increasing function  $\phi$  is equal to  $p(\hat{c}, f)$ , for example with a simple bisection method.

## 7. EXPERIMENTS

To evaluate the performance of robustness quantification and compare it to uncertainty quantification, we use both methods to assess the reliability of the predictions issued by a simple Naive Bayes Classifier. The class has three possible values, and the four features can take 2, 3, 3 and 4 possible values, respectively. To be able to study the effect of distribution shift and limited data, we generate our own data so we can control both aspects.

**7.1. The test set.** Our test distribution  $P_{\text{test}}$  is a mixture of two distributions:  $P_{\text{test}} = (1 - \beta)P_{\text{fix}} + \beta P_{\text{rand}}$ . In this mixture,  $P_{\text{fix}}$  is a fixed distribution that satisfies the Naive Bayes assumption (Equation (17)). The class probabilities are 0.4, 0.35, and 0.25. For each class value  $c$  and feature  $F_i$ , we assign probability 0.85 to one of the feature values and distribute the rest of the probability mass uniformly over the other feature values. This part of the mixture gives structure to  $P_{\text{test}}$ , creating a correlation between the features and class that makes classification possible.  $P_{\text{rand}}$  is a randomly generated distribution, obtained by generating a random number for each  $(c, f)$  and then normalizing so they sum to one. This part of the mixture makes sure that  $P_{\text{test}}$  does not satisfy the Naive Bayes assumption, and that the classification task is sufficiently difficult. We don't analyze the effect of  $\beta$  in our experiments; it has a fixed value  $\beta = 0.3$  throughout. The same is true for  $P_{\text{rand}}$ : while it is random, we keep it fixed throughout all experiments. This implies that  $P_{\text{test}}$  is fixed as well. The test set  $D_{\text{test}}$  is a random sample of size  $N_{\text{test}} = 1000$  according to  $P_{\text{test}}$ .

**7.2. The training sets.** Our training distributions are a mixture of  $P_{\text{test}}$  and a random distribution  $P_{\text{shift}}$ , with  $\gamma \in [0, 1]$  as mixture coefficient:  $P_{\text{train}} = (1 - \gamma)P_{\text{test}} + \gamma P_{\text{shift}}$ . The maximum amount of distribution shift is determined by  $\gamma$ , but the actual amount of distribution shift also depends on the random distance between  $P_{\text{shift}}$  and  $P_{\text{test}}$ . We use several values of  $\gamma$  in our experiments to study the

effect of distribution shift and consider multiple instantiations of  $P_{\text{shift}}$  to make sure that there is some actual distribution shift present. For each  $P_{\text{train}}$  obtained in this way, the corresponding training sets  $D_{\text{train}}$  are random samples of size  $N_{\text{train}}$ . To study the effect of limited data, our experiments consider several values of  $N_{\text{train}}$ .

**7.3. A single experiment.** For a single training set  $D_{\text{train}}$  of size  $N_{\text{train}}$ , the experiment we conduct goes as follows. First, we use  $D_{\text{train}}$  to learn an NBC. Do to so, we begin by using 5-fold cross validation to optimize the smoothing parameter  $\alpha$  for maximal accuracy. Next, we use this optimal  $\alpha$  to learn a final NBC based on the whole training set. With the obtained NBC, for each of the 1000 instances in  $D_{\text{test}}$ , we then make a prediction  $\hat{c}$  based on the features  $f$ , and compute the uncertainty metrics  $u_m(f)$ ,  $u_H(f)$ ,  $u_a(f)$ ,  $u_i(f)$  and  $u_e(f)$  and robustness metrics  $\varepsilon_{\text{glob}}(f)$  and  $\varepsilon_{\text{loc}}(f)$  for that prediction.

**7.4. Accuracy-acceptance curves.** To evaluate the performance of each of these metrics, for a single  $D_{\text{train}}$  and its corresponding NBC, we look at how good they are at assessing the reliability of the test instances. To that end, we use each metric to order the 1000 instances in  $D_{\text{test}}$ . For the uncertainty metrics, we do this in order of increasing uncertainty. For the robustness metrics, we do this in order of decreasing robustness. For each of these orderings, the hope is now of course that the first (least uncertain, most robust) instances are the most reliable ones, whereas the last (most uncertain, least robust) instances are the least reliable. In order to evaluate this, we chose to use *accuracy-acceptance curves*.<sup>2</sup> For each metric, such a curve plots the accuracy of the predictions of the NBC for the first  $N$  instances in the ordering provided by the metric (so the accuracy for the  $N$  most reliable instances, one would hope), as a function of the so-called acceptance rate  $r = N/N_{\text{test}} = N/1000$ . For  $r = 1$ , this yields the accuracy on the entire test set, and hence the same result for all metrics. For  $r < 1$ , good metrics are able to increase this accuracy. So better metrics yield higher accuracy-acceptance curves.

**7.5. Experimental setup.** We explore the influence of limited data and distribution shift by considering different sizes of training sets and different amounts of distribution shift. The considered values for  $N_{\text{train}}$  are  $\{25, 50, 100\}$  and the values for  $\gamma$  are  $\{0, 0.2, 0.4\}$ .

For each combination of  $N_{\text{train}}$  and  $\gamma$  we generate 10 different training distributions  $P_{\text{train}}$  (with the same  $\gamma$  and 10 different random  $P_{\text{shift}}$ ) and use each of these to sample 10 different training sets  $D_{\text{train}}$  of size  $N_{\text{train}}$ . For each of the resulting 100 training sets, we then run the

<sup>2</sup>These are very similar to the so-called accuracy-rejection curves [16] that are commonly used to compare classifiers with a reject option: the only difference is that we plot the accuracy as a function of the acceptance rate instead of the rejection rate. We prefer our version because it emphasizes that we are interested in the accuracy on the accepted (more reliable) instances.



experiment described in Section 7.3 and, for each of the considered metrics, construct an accuracy-acceptance curve as described in Section 7.4.

To evaluate the overall performance of a metric, we consider the average of the 100 obtained accuracy-acceptance curves. To evaluate to variability of this performance (to which extent the 100 training sets yield different results), we consider the standard-deviation. The results are depicted in Figure 1 and 2, respectively.

**7.6. Results.** In Figure 1, we clearly see that the smaller the training set is, so comparing the plots from top to bottom, the worse the uncertainty metrics perform, since those curves get lower and lower, while the curves of the two robustness metrics stay relatively high. A similar conclusion can be drawn for distribution shift. The more distribution shift there is, so if we move towards the right, the worse the performance of the uncertainty metrics is in comparison to the robustness measures, especially so for small training sets. The performance of the robustness metrics also seems to be the least affected by distribution shift or limited data, dropping considerably only for the most extreme case (on the bottom right). Meanwhile, even without distribution shift and for larger training sets, our robustness metrics are competitive with all the uncertainty metrics. Comparing our two robustness metrics, we see that their performance is similar, with  $\epsilon_{\text{loc}}$  performing slightly better for small data sets without distribution shift, and  $\epsilon_{\text{glob}}$  performing better in the face of both challenges.

Looking at Figure 2, which displays the standard deviations, it becomes clear that our robustness metrics not only perform better in terms of overall average performance, but that their performance is also more stable, in the sense that it varies considerably less across the different training sets.

## 8. CONCLUSION/DISCUSSION

The main conclusion of our contribution is that the robustness metrics that are provided by robustness quantification do a good job at indicating to which extent a prediction of a generative classifier is reliable. We furthermore see that robustness quantification is competitive with uncertainty quantification in scenarios with sufficiently large training sets and no distribution shift, and that it outperforms it in the face of both challenges.

So while robustness quantification ignores uncertainty, quantifying only how much uncertainty we could cope with if it were present, this seems to work better than quantifying epistemic uncertainty, or quantifying aleatoric uncertainty in the face of epistemic uncertainty.

Nevertheless, in our future work, since it seems to us that robustness and uncertainty metrics quantify entirely different aspects of reliability, we'd like to explore to which extent they can be combined to arrive at even better reliability metrics. To that end, we'd also first like

to gain a better understanding about why, and in which contexts, robustness quantification works well.

Finally, we would of course like to confirm our conclusions with more extensive experiments, for classifiers with more complex (deep) model architectures and continuous features, and for real classification problems based on collected rather than generated data.

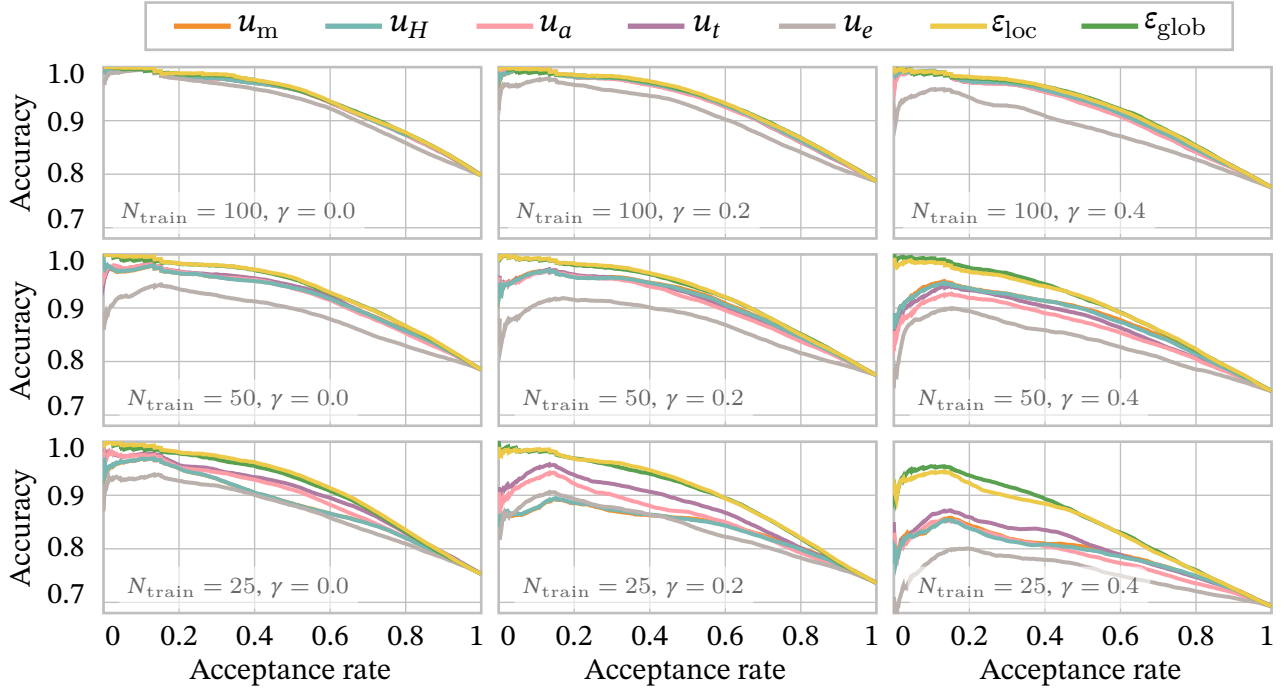
## ADDITIONAL AUTHOR INFORMATION

**Acknowledgements.** We would like to thank the anonymous reviewers for their time, thorough reading and, especially, for their constructive and helpful feedback. The work of both authors was partially supported by Ghent University's Special Research Fund, through Jasper De Bock's starting grant number 01N04819.

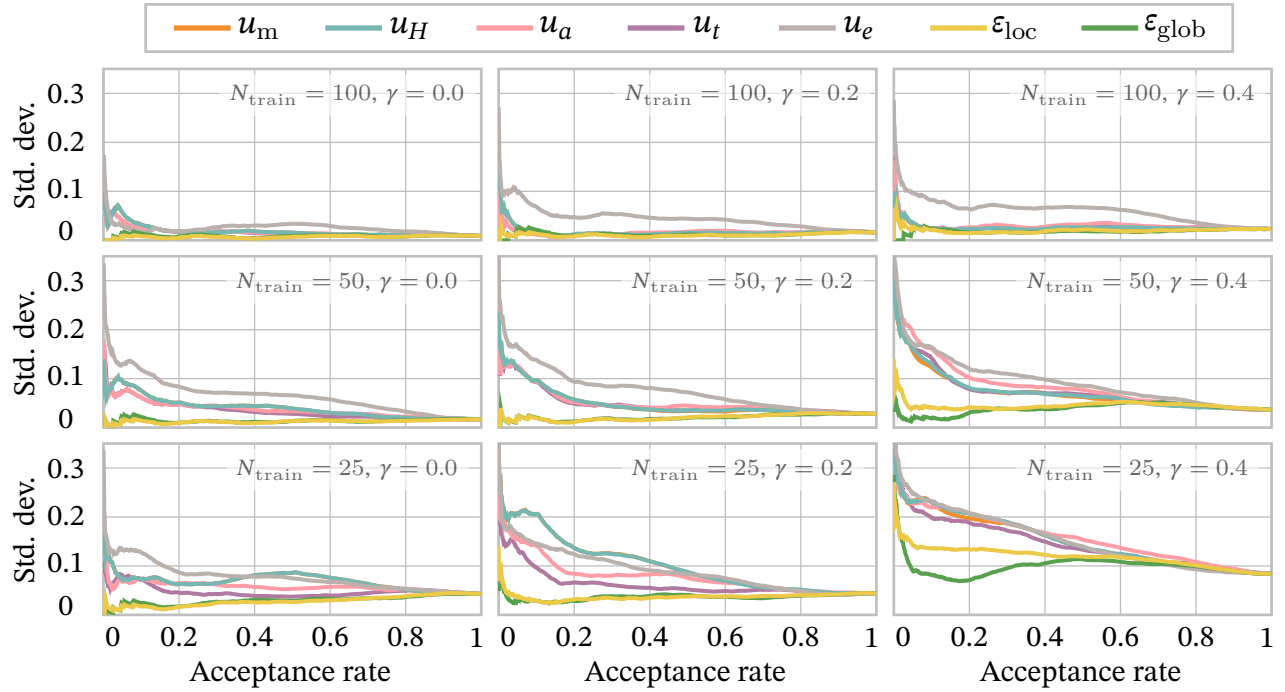
**Author contributions.** Both authors contributed to the conceptualization of the ideas in this paper. The first author ran the experiments and wrote a first version of the paper, which then underwent a series of revisions under the supervision of the second author.

## REFERENCES

- [1] Thomas Augustin, Frank P.A. Coolen, Gert De Cooman, and Matthias C.M. Troffaes. *Introduction to imprecise probabilities*. John Wiley & Sons, 2014. DOI: [10.1002/9781118763117](https://doi.org/10.1002/9781118763117).
- [2] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. "Recent advances in adversarial training for adversarial robustness". In: *arXiv preprint arXiv:2102.01356* (2021).
- [3] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013. DOI: [10.1007/978-1-4757-4286-2](https://doi.org/10.1007/978-1-4757-4286-2).
- [4] Nicholas Carlini et al. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).
- [5] Giorgio Corani, Joaquín Abellán, Andrés Masegosa, Serafin Moral, and Marco Zaffalon. "Classification". In: *Introduction to Imprecise Probabilities*. John Wiley & Sons, Ltd, 2014. Chap. 10, pp. 230–257. DOI: [10.1002/9781118763117.ch10](https://doi.org/10.1002/9781118763117.ch10).
- [6] Alvaro H. C. Correia, Robert Peharz, and Cassio P. de Campos. *Towards Robust Classification with Deep Generative Forests*. 2020. arXiv: [2007.05721](https://arxiv.org/abs/2007.05721) [stat.ML].



**Figure 1.** The means of the accuracy-acceptance curves for decreasing  $N_{\text{train}}$  (100, 50, 35 from top to bottom) and increasing  $\gamma$  (0, 0.2, 0.4 left to right).



**Figure 2.** The standard deviations of the accuracy-acceptance curves for decreasing  $N_{\text{train}}$  (100, 50, 25 from top to bottom) and increasing  $\gamma$  (0, 0.2, 0.4 left to right).

- [7] Jasper De Bock, Cassio P. de Campos, and Alessandro Antonucci. “Global Sensitivity Analysis for MAP Inference in Graphical Models”. In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2014/file/0966289037ad9846c5e994be2a91bafa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2014/file/0966289037ad9846c5e994be2a91bafa-Paper.pdf).
- [8] Pedro Domingos and Michael Pazzani. “On the Optimality of the Simple Bayesian Classifier under Zero-One Loss”. In: *Machine Learning* 29 (1997), pp. 103–130. DOI: [10.1023/A:1007413511361](https://doi.org/10.1023/A:1007413511361).
- [9] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973. DOI: [10.2307/2344977](https://doi.org/10.2307/2344977).
- [10] Jerome H. Friedman. “On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality”. In: *Data Mining and Knowledge Discovery* 1 (1997), pp. 55–77. DOI: [10.1023/A:1009778005914](https://doi.org/10.1023/A:1009778005914).
- [11] A Gelman et al. *Bayesian Data Analysis*. CRC Press, 2013. DOI: [10.1201/b16018](https://doi.org/10.1201/b16018).
- [12] Eyke Hüllermeier, Sébastien Destercke, and Mohammad Hossein Shaker. “Quantification of Credal Uncertainty in Machine Learning: A Critical Analysis and Empirical Comparison”. In: vol. 180. *Proceedings of Machine Learning Research*. PMLR, 2022, pp. 548–557. URL: <https://proceedings.mlr.press/v180/hullermeier22a.html>.
- [13] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods”. In: *Machine learning* 110.3 (2021), pp. 457–506. DOI: [10.1007/s10994-021-05946-3](https://doi.org/10.1007/s10994-021-05946-3).
- [14] Pang Wei Koh et al. “WILDS: A Benchmark of in-the-Wild Distribution Shifts”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. *Proceedings of Machine Learning Research*. PMLR, 2021, pp. 5637–5664. URL: <https://proceedings.mlr.press/v139/koh21a.html>.
- [15] Denis D. Mauá, Fabio G. Cozman, Diarmaid Conaty, and Cassio P. Campos. “Credal Sum-Product Networks”. In: vol. 62. *Proceedings of Machine Learning Research*. PMLR, 2017, pp. 205–216. URL: <https://proceedings.mlr.press/v62/mau%C3%A117a.html>.
- [16] Malik Sajjad Ahmed Nadeem, Jean-Daniel Zucker, and Blaise Hanczar. “Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option”. In: *Machine Learning in Systems Biology*. PMLR, 2009, pp. 65–81.
- [17] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009. DOI: [10.5555/1462129](https://doi.org/10.5555/1462129).
- [18] Sarunas J. Raudys, Anil K. Jain, et al. “Small sample size effects in statistical pattern recognition: Recommendations for practitioners”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.3 (1991), pp. 252–264. DOI: [10.1109/34.75512](https://doi.org/10.1109/34.75512).
- [19] Yusuf Sale et al. *Label-wise Aleatoric and Epistemic Uncertainty Quantification*. 2024. arXiv: [2406.02354](https://arxiv.org/abs/2406.02354).
- [20] Mohammad Hossein Shaker and Eyke Hüllermeier. “Aleatoric and epistemic uncertainty with random forests”. In: *Advances in Intelligent Data Analysis XVIII*. Springer, 2020, pp. 444–456. DOI: [10.1007/978-3-030-44584-3\\_35](https://doi.org/10.1007/978-3-030-44584-3_35).
- [21] Rohan Taori et al. “Measuring Robustness to Natural Distribution Shifts in Image Classification”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18583–18599. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf).
- [22] Andrius Vabalas, Emma Gowen, Ellen Poliakoff, and Alexander J. Casson. “Machine learning algorithm validation with a limited sample size”. In: *PloS one* 14.11 (2019), pp. 1–20. DOI: [10.1371/journal.pone.0224365](https://doi.org/10.1371/journal.pone.0224365).
- [23] Marco Zaffalon. “The naive credal classifier”. In: *Journal of Statistical Planning and Inference* 105.1 (2002), pp. 5–21. DOI: [10.1016/S0378-3758\(01\)00201-4](https://doi.org/10.1016/S0378-3758(01)00201-4).
- [24] John R. Zech et al. “Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study”. In: *PLoS medicine* 15.11 (2018), pp. 1–17. DOI: [10.1371/journal.pmed.1002683](https://doi.org/10.1371/journal.pmed.1002683).