A probability transducer
for machine-learning classifiers

K. Dyrland
<kjetil.dyrland@gmail.com>

A. S. Lundervold [†]
<alexander.selvikvag.lundervold@hvl.no>

P.G.L. Porta Mana
<pgl@portamana.org>

(listed alphabetically)

Dept of Computer science, Electrical Engineering and Mathematical Sciences
Western Norway University of Applied Sciences, Bergen, Norway
[†] & Mohn Medical Imaging and Visualization Centre, Bergen, Norway

Draft. 14 April 2022; updated 28 May 2022

✎ The present work sells probabilities for machine-learning algorithms at a bargain price. It shows that these probabilities are just what those algorithms need in order to increase their sales.

1 The inadequacy of common classification approaches

As the potential of using machine-learning algorithms in important fields such as medicine or drug discovery increases, the machine-learning community ought to keep in mind what the actual needs and inference contexts in such fields are. We must avoid trying (consciously or unconsciously) to convince such fields to change their needs or ignore their contexts just to fit machine-learning solutions that are available and fashionable at the moment. Rather, we must make sure the that solutions fit needs & context, and amend them if they do not.

The machine-learning mindset and approach to problems such as classification in such new important fields is often still inadequate in many respects. It reflects simpler needs and contexts of many early inference problems successfully tackled by machine learning.

A stereotypical 'cat vs dog' image classification, for instance, has four very important differences from a 'disease *I* vs disease *II*' medical classification, or from an 'active vs inactive' drug classification:

This document is designed for screen reading and two-up printing on A4 or Letter paper

(i) Nobody presumably dies or loses large amounts of money if a cat image is misclassified as dog or vice versa. But a person can die if a disease is misdiagnosed; huge capitals can be lost if an ultimately ineffective drug candidate is pursued. The *gains and losses* – or generally speaking the *utilities* – of correct and incorrect classifications in the former problem and in the two latter problems are vastly different.

(ii) To what purpose do we try to guess whether an image's subject is a cat or a dog? For example because we must decide whether to put it in the folder 'cats' or in the folder 'dogs'. To what purpose do we try to guess a patient's disease or a compound's chemical activity? A clinician does not simply tell a patient "You probably have such-and-such disease. Goodbye!", but has to decide among many different kinds of treatments. The candidate drug compound may be discarded, pursued as it is, modified, and so on. *The ultimate goal of a classification is always some kind of decision*, not just a class guess. In the cat-vs-dog problem there is a natural one-one correspondence between classes and decisions. But in the medical or drug-discovery problems *the set of classes and the set of decisions are very different*, and have even different numbers of elements.

(iii) If there is a 70% probability that an image's subject is a cat, then it is natural to put it in the folder 'cats' rather than 'dogs' (if the decision is only between these two folders). If there is a 70% probability that a patient has a particular health condition, it may nonetheless be better to dismiss the patient – that is, to behave as if there was no condition. This is the optimal decision, for example, when the only available treatment for the condition would severely harm the patient if the condition were not present. Such treatment would be recommended only if the probability for the condition were much higher than 70%. Similarly, even if there is a 70% probability that a candidate drug is active it may nonetheless be best to discard it. This is the economically most advantageous choice if pursuing a false-positive leads to large economic losses. *The target of a classification is not what's most probable, but what's optimal.* ✎ add something on proper probabilities, connecting to (iv)

(iv) The relation from image pixels to house-pet subject may be almost deterministic; so we are effectively extrapolating or looking for a function pet = f(pixels) contaminated by little noise. But the relation between medical-test scores or biochemical features on one side, and disease or drug activity on the other, is typically *probabilistic*; so a

function disease $= f$ (scores) or activity $= f$ (features) does not even exist. *We are assessing statistical relationships* P(disease, scores) or P(activity, features) instead, which include deterministic ones as special cases.

In summary, there is place to improve classifiers so as to (i) quantitatively take into account actual utilities, (ii) separate classes from decisions, (iii) target optimality rather than 'truth', (iv) output and use proper probabilities.

In artificial intelligence and machine learning it is known how to address all these issues in principle – the theoretical framework is for example beautifully presented in the first 18 chapters or so of Russell & Norvig's (2022) text 🔧 add also refs to Cheeseman.

Issues (i)–(iii) are simply solved by adopting the standpoint of *Decision Theory*, which we briefly review in § **??** and we discuss at length in our companion work[1]. In short: The set of decisions and the set of classes pertinent to a problem are separated if necessary. A utility is associated to each decision, relative to the occurrence of each particular class; these utilities are assembled into a utility matrix: one row per decision, one column per class. This matrix is multiplied by a column vector consisting in the probability for the classes. The resulting vector of numbers contains the expected utility of each decision. Finally we select the decision having *maximal expected utility*, according to the principle bearing this name. Such procedure also takes care of the class imbalance problem; cf. the analysis by Drummond & Holte 2005 (they use the term 'cost' instead of 'utility').

Clearly this procedure is computationally inexpensive and ridiculously easy to implement in any machine-learning classifier. The difficulty is that this procedure requires *sensible probabilities* for the classes, which brings us to issue (iv), the most difficult.

Some machine-learning algorithms for classification, such as support-vector machines, output only a class label. Others, such as deep networks, output a set of real numbers that can bear some qualitative relation to the plausibilities of the classes. But these numbers cannot be reliably interpreted as proper probabilities, that is, as the degrees of belief assigned to the classes by a rational agent[2]; or, in terms of 'populations'[3], as the expected frequencies of the classes in the hypothetical population of

---

[1] Dyrland et al. 2022a.   [2] MacKay 1992a; Gal & Ghahramani 2016; Russell & Norvig 2022 chs 2, 12, 13.   [3] Lindley & Novick 1981; Fisher 1967 § II.4.

units (degrees of belief and frequencies being related by de Finetti's theorem[4]). Algorithms that internally do perform probabilistic calculations, for instance naive-Bayes or logistic-regression classifiers[5], unfortunately rest on probabilistic assumptions, such as independence and particular shapes of distributions, that are often unrealistic (and their consistency with the specific application is rarely checked). Only particular classifiers such as Bayesian neural networks[6] output sensible probabilities, but they are computationally very expensive. The stumbling block is the extremely high dimensionality of the feature space, which makes the calculation of the probabilities

$$P(\text{class, feature} \mid \text{training data})$$

(a problem opaquely called 'density regression' or 'density estimation'[7]) computationally unfeasible.

If we solved the issue of outputting proper probabilities then the remaining three issues would be easy to solve, as discussed above.

In the present work we propose an alternative solution to calculate proper class probabilities, which can then be used in conjunction with utilities to perform the final classification or decision.

This solution consists in a sort of 'transducer' or 'calibration curve' that transforms the algorithm's raw output into a probability. It has a low computational cost, can be applied to all commonly used classifiers and even to simple regression algorithms, does not need any changes in algorithm architecture or in training procedures, and is grounded on first principles.

Moreover, this transducer has two great benefits. First, it allows us to calculate both the probability of class conditional on features, and *the probability of features conditional on class*. In other words it allows us to use the classification algorithm in both 'discriminative' and 'generative' modes[8], even if the algorithm was not designed for a generative use. Second, its computation automatically also yields a 'probability of the probability', or more precisely a quantification of how much the probability would change if we had further training data.

---

[4] Bernardo & Smith 2000 ch. 4; Dawid 2013.   [5] Murphy 2012 § 3.5, ch. 8; Bishop 2006 §§ 8.2, 4.3; Barber 2020 ch. 10, § 17.4.   [6] Neal & Zhang 2006; Bishop 2006 § 5.7.   [7] Ferguson 1983; Thorburn 1986; Hjort 1996; Dunson et al. 2007.   [8] Russell & Norvig 2022 § 21.2.3; Murphy 2012 § 8.6.

The probability thus obtained is also combined with utilities to perform the final classification task.

We show on a concrete dataset that this solution always leads to an improvement in classification, in some cases with a relative increase as high as 30%.

The main idea is explained in § 2. Section🖉 ... explains how the resulting probabilities are combined with utilities to perform classification. In §🖉 ... we apply the transducer to🖉 ...

🔧 finish synopsis

2　An output-to-probability transducer

In the present section we explain the main idea in informal and intuitive terms, and give its mathematical essentials only. Stricter logical derivation and more mathematical details are left to appendix🖉 ... .

2.1　Main idea: algorithm output as a proxy for the features

Let us consider first the essentials behind a classification (or regression) problem. We have the following quantities:

- the *feature* values of a set of known units,
- the *classes* of the same set of units,

which together form our *training data*; and

- the feature value of a new unit,

where the 'units' can be widgets, images, patients, drug compounds, and so on, depending on the classification problem. From these quantities we would like to infer the class of the new unit. This inference consists in probability values (we omit 'new' in equations for brevity)

$$\mathrm{P}(\text{class of unit} \mid \text{feature of unit, classes \& features of training data}) \quad (1)$$

for each possible class.

These probabilities are obtained through the rules of the probability calculus[9]; in this case specifically through the so-called de Finetti theorem[10] which connects training data and new unit.

---

[9] Jaynes 2003; Russell & Norvig 2022 chs 12–13; Gregory 2005; Hailperin 2011; Jeffreys 1983. [10] Bernardo & Smith 2000 ch. 4; Dawid 2013.

Combined with a set of utilities, these probabilities allow us to determine an optimal, further decision to be made among a set of alternatives. Note that the inference (1) includes deterministic interpolation, i.e. the assessment of a function class = f(feature), as a special case, when the probabilities are essentially 0s and 1s.

A trained classifier should ideally output the probabilities above when applied to the new unit. Machine-learning classifiers trade this capability for computational speed – with an increase in the latter of several orders of magnitude[11]. Thus their output cannot be considered a probability, but *it still carries information about both class and feature variables*.

Now we acknowledge the fact that the information contained in the feature and in the training data, relevant to the class of the new unit, is simply inaccessible to us because of computational limitations. We do have access to the output for the new unit, however, which does carry relevant information. Thus want to calculate a probability such as P(class of unit | output for unit).

To calculate this probability, however, it is necessary to have examples of further pairs (class of unit, output for unit), of which the new unit's pair can be considered a 'representative sample'[12] and vice versa – exactly for the same reason why we need training data in the first place.

For this purpose, can we use the pairs (class of unit, output for unit) of the training data? This would be very convenient, as those pairs are readily available. But answer is no. The reason is that the outputs of the training data are produced from the features *and the classes* jointly; this is the very point of the training phase. There is therefore a direct informational dependence between the classes and the outputs of the training data. For the new unit, on the other hand, the classifier produces its output from the feature alone. The new unit is not a representative sample of the training data.

We need a data set where the outputs are generated by simple application of the algorithm to the feature, as it would occur in its concrete use, and the classes are known. The *test data* of standard machine-learning procedures are exactly what we need. The new unit

---

[11] to understand this trade-off in the case of neural-network classifiers see e.g. MacKay 1992b,c,a; Murphy 2012 § 16.5 esp. 16.5.7; see also the discussion by Self & Cheeseman 1987. [12] for a critical analysis of the sometimes hollow term 'representative sample' see Kruskal & Mosteller 1979a,b,c; 1980.

can be considered a representative sample of the test data, and vice versa. We call such data 'transducer-calibration data', owing to its specific purpose.

The probability we arrive at is therefore

$$P(\text{class of unit} \mid \text{output for unit, classes \& outputs for calibration data}) . \quad (2)$$

For algorithms that yield an output much simpler than the features, such as a vector of few real components, the probability above can be exactly calculated. Thus, once we obtain the classifier's output for the new unit, we can calculate a probability for the new unit's class.

The idea above can also be informally understood in two ways. First: the classifier's output is regarded as a proxy for the feature. Second: the classifier is regarded as something analogous to a *diagnostic test*, such as any common diagnostic or prognostic test used in medicine for example. We do not take diagnostic-test results at face value – if a flu test is 'positive' we do not conclude that the patient has the flu – but rather arrive at a probability that the patient has the flu, given some statistics about results of tests performed on 'verified samples' of true-positive and true-negative patients[13].

The probability values (4) for a fixed class and variable output give us a sort of 'calibration curve' (or calibration hypersurface for multidimensional outputs) of the output-to-probability transducer for the classifier. See fig.✎ ... as an example. It must be stressed that such curve needs to be calculated only once, and it can be used for all further applications of the classifier to new units.

What is the relation between the ideal incomputable probability (1) and the probability (4) obtained by proxy? It can be shown that the two are related by convex combination or mixing:

$$P(\text{class} \mid \text{output, classes \& outputs for calibration data}) =$$

$$\sum_{\substack{\text{feature} \\ \text{training data}}} w(\text{feature, training data}) \times P(\text{class} \mid \text{feature, training data}) \quad (3)$$

where w are positive and normalized functions. This means that the proxy probability is generally less sharp, that is, farther away from 0

---

[13] Sox et al. 2013 ch. 5; Hunink et al. 2014 ch. 5; see also Jenny et al. 2018.

and 1, than the ideal one. This is an obvious consequence of the loss of information about the feature value. Such mixing, on the other hand, only leads to more conservative probabilities, not to over-confident ones.

✎ add note about the fact that the goodness of the results still greatly depends on the goodness of the classifier.

🔧 add that *no new test set is needed*! The calibration set also works as 'test set' thanks to the probability calculus

2.2   Calculation of the probabilities

Let us denote by c the class value of a new unit, by y the output of the classifier for the new unit, and by $D := \{c_i, y_i\}$ the classes and classifier outputs for the transducer-calibration data.

It is more convenient to focus on the *joint* probability of class and output given the data,

$$p(c, y \mid D),\qquad(4)$$

rather than on the conditional probability of the class given the output and calibration data, (4).

The joint probability is calculated using standard non-parametric Bayesian methods[14]. 'Non-parametric' in this case means that we do not make any assumptions about the shape of the probability curve as a function of c, y (contrast this with logistic regression, for instance), or about special independence between the variables (contrast this with naive-Bayes). The only assumption made – and we believe it is quite realistic – is that the curve must have some minimal degree of smoothness. This assumption allows for much leeway, however: fig.✎ … for instance shows that the probability curve can still have very sharp bends, as long as they are not cusps.

Non-parametric methods differ from one another in the kind of 'coordinate system' they select on the infinite-dimensional space of all possible probability curves, that is, in the way they represent a general positive normalized function.

We choose the representation discussed by Dunson & Bhattacharya[15]. The end result of interest in the present section is that the probability

---

[14] for introductions and reviews see e.g. Walker 2013; Müller & Quintana 2004; Hjort 1996.

[15] Dunson & Bhattacharya 2011; see also the special case presented by Rasmussen 1999.

density $p(c, y \mid D)$, with c discrete and y continuous and possibly multidimensional, is expressed as a sum

$$p(c, y \mid D) = \sum_k q_k \, A(c \mid \alpha_k) \, B(y \mid \beta_k) \tag{5}$$

of a finite but large number of terms. Each term is the product of a positive weight q_k, a probability distribution $A(c \mid \alpha_k)$ for c depending on parameters α_k, and a probability density $B(c \mid \beta_k)$ for y depending on parameters β_k. The parameter values can be different from term to term, as indicated by the index k. The weights $\{q_k\}$ are normalized.

This mathematical representation can approximate (under some norm) any smooth probability density in c and y. It has the advantages of being automatically positive and normalized, and of readily producing the marginal distributions for c and for y:

$$p(c) = \sum_k q_k \, A(c \mid \alpha_k) \,, \qquad p(y) = \sum_k q_k \, B(y \mid \beta_k) \,, \tag{6}$$

from which also the conditional distributions are easily obtained:

$$p(c \mid y) = \sum_k \frac{q_k \, B(y \mid \beta_k)}{\sum_l q_l \, B(y \mid \beta_l)} \, A(c \mid \alpha_k) \,, \tag{7a}$$

$$p(y \mid c) = \sum_k \frac{q_k \, A(c \mid \alpha_k)}{\sum_l q_l \, A(c \mid \alpha_l)} \, B(y \mid \beta_k) \,. \tag{7b}$$

These will be important in the following discussion.

The parametric distributions $A(c \mid \alpha)$ and $B(y \mid \beta)$ are chosen by us according to convenience. For the first we use a simple categorical distribution with parameters α. In a binary-classification case, where the class variable c takes on conventional values $\{0, 1\}$, it is

$$A(c \mid \alpha) = c \, \alpha + (1 - c) \, (1 - \alpha) \equiv \begin{cases} \alpha & \text{if } c = 1, \\ 1 - \alpha & \text{if } c = 0. \end{cases} \tag{8}$$

For the second we use a Gaussian distribution, $\beta \equiv (\mu, \sigma)$ being its mean and standard deviation:

$$B(y \mid \beta) = \mathrm{N}(y \mid \mu, \sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y - \mu)^2}{2\sigma^2}\right], \tag{9}$$

or a product of such Gaussians, each with its own parameters, if y is multidimensional. If the classifier's output y is a bounded variable we

perform a transformation of its range onto the real line first (taking due care of Jacobian determinants for density transformations).

The weights $\{q_k\}$ and the parameters $\{\alpha_k\}$, $\{\beta_k\}$ are the heart of this representation, because the shape of the probability curve $p(c \mid y, D)$ depends on their values. They are determined by the test data D. Their calculation is done via Markon-chain Monte Carlo sampling, which we discuss in appendix✏ …. For low-dimensional y and discrete c (or even continuous, low-dimensional c, which means we are working with a regression algorithm), this calculation can be done in a matter of hours, and *it only needs to be done once*.

Once calculated, these parameters are saved in memory and can be used to compute any of the probabilities (5), (6), (7) as needed, as discussed in the next subsection. Such computations take less than a second.

Note that the role of the classifier in this calculation is simply to produce the outputs y for the calibration data, after having been trained in any standard way on a training data set. No changes in its architecture or in its training procedure have been made, nor are any required.

2.3 Applying the probability transduction: discriminative and generative modes

Let us now discuss the application of the various probability functions presented above. For simplicity we shall omit the dependence '$\ldots, D$' on the calibration data, leaving it implicitly understood.

We have a new unit, which could be part of an evaluation test-set or coming from a real application after deployment. Its features are fed to the classifier, which outputs the real value y. We can then proceed in two ways:

Discriminative mode: *probability of class given output*
  We calculate $p(c \mid y)$ from formula (7a), for each value of c, say $c = 0$ and $c = 1$ in a binary-classification case. These are the probabilities of the classes.

The discriminative mode is the standard way of proceeding in classification and does not need further discussion. We have simply translated the classifier's raw output into a more sensible probability. From this point of view the function $p(c \mid y)$ can be considered as a more appropriate

substitute of the softmax function, for instance, at the output of a neural network.

Generative mode: *probability of class given output and base rates*

We calculate $p(y \mid c)$ from formula (7b), again for each value of c. The probability of each class c is then obtained through Bayes's theorem after supplying the *population prevalence* r_c of the class:

$$p(c \mid y, \text{prevalences}) = \frac{p(y \mid c)\, r_c}{\sum_c p(y \mid c)\, r_c} \ . \tag{10}$$

The population prevalences[16], also called *base rates*[17], are the relative frequencies of occurrence of the various classes in the population whence our unit originates. This notion is very familiar in medicine and epidemiology. For example, a particular type of tumour can have a prevalence of 0.01% among people of a given age and sex, meaning that 1 person in 10 000 among them has that kind of tumour, as obtained through a large survey.

🔧 maybe move parts below to independent section

The generative mode, generally not available for classifiers with a discriminative design, is the required way to calculate the class probabilities if the calibration set does not have the same class frequencies as the population on which the classifier will be employed – let us call the latter the 'real population'. For example, two classes may appear in a 50%/50% proportion in the calibration set but in a 90%/10% proportion in the real population. This frequency discrepancy can occur for several reasons. Examples: samples of the real population are unavailable or too expensive to be used for calibration purposes; the class statistics of the real population has suddenly changed right after the deployment of the classifier; it is necessary to use the classifier on a slightly different population; the sampling of calibration data was poorly designed.

2.4   The double role of the calibration data set

In the preceding analysis we discussed why the construction of the probability transducer requires a calibration set separate from the training & validation set. Do we then need a third, separate 'test set' for the final evaluation and comparison of candidate classifiers or hyperparameters?

---

[16] Sox et al. 2013 ch. 3; Hunink et al. 2014 § 5.1.   [17] Bar-Hillel 1980; Axelsson 2000.

This would be inconvenient: it would reduce the amount of data available for training.

The answer is no: *the calibration set automatically also serves as a test set*.

It may be useful to explain why this is the case, especially for those who may take for granted the universal necessity of a test set.

Many standard machine-learning methodologies need a test set because they are only an approximation of ideal inference performed with the probability calculus. The latter needs no division of available data into different sets: something analogous to such division is automatically made internally, so to speak. It can be shown[18] that the mathematical operations behind the probability rules correspond to making *all possible* divisions of available data between 'training' and 'test', as well as all possible cross-validations with folds of all orders. This is the reason why ideal inference by means of the probability calculus is almost computationally impossible in some cases, and we have to resort to approximate but faster machine-learning methods. The latter typically do not do such data partitions automatically; the latter need to be made – and only approximately – by hand.

✎ explain how testing is done

3   Utility-based classification

We refer to our companion work Dyrland et al. 2022a, § 2, for a more detailed presentation of decision theory and for references. In the following we assume familiarity with the material presented there.

Our classification or decision problem has a set of decisions, which we can index by $i = 1, 2, \ldots$. As discussed in § 1, these need not be the same as the possible classes; the two sets may even be different in number. But the true class, which is unknown, determines the *utility* that a decision yields. If we choose decision i and the class c is true, our eventual utility will be U_{ic}.[19] These utilities are assembled into a rectangular matrix (U_{ic}) with one row per decision and one column per class. Note that the case where decisions and classes are in a natural one-one correspondence, as in the cat-vs-dog classification example of § 1, is just a particular case of this more general point of view. In such

---

[18] Porta Mana 2019; Fong & Holmes 2020; Wald 1949; many examples of this fact are scattered across the text by Jaynes 2003.   [19] We apologize for the difference in notation from our companion work, where the class variable is 'j' and the utilities 'U_{ij}'

a specific case we may replace 'decision' with 'class' in the following discussion, and the utility matrix is square.

Denote by (p_c) the probabilities for the classes, calculated as described in § 2.3 above, and collect them into a column vector.

The expected utility $\bar{U}_i$ of decision i is then given by the product of the matrix (U_{ic}) and the column vector (p_c):

$$\bar{U}_i := \sum_c U_{ic}\, p_c \, . \tag{11}$$

Finally, according to the principle of maximum expected utility, we choose the decision i^* having largest $\bar{U}_i$:

$$\text{choose} \quad i^* = \arg\max_i\{\bar{U}_i\} \equiv \arg\max_i\left\{\sum_c U_{ic}\, p_c\right\} . \tag{12}$$

The matrix multiplication and subsequent selection are computationally inexpensive; they can be considered as substitutes of the 'argmax' selection that typically happen at the continuous output of a classifier.

4   Demonstration and results

4.1   Overview

We illustrate the implementation of the probability transducer and its combination with utility-based decisions in a concrete example. The evaluation of the results is also made from the standpoint of decision theory, using utility-based metrics, as explained in our companion paper[20].

A couple of remarks may clarify the purpose of this illustration and our choice of classification problem.

The internal consistency of decision theory guarantees that utility-based decisions always improve on, or at least give as good results as, any other procedure, including the standard classification procedures used in machine learning. This is intuitively obvious: we are, after all, grounding our single class choices upon the same gains & losses that underlie our classification problem and that are used in its evaluation. The present illustration is therefore not a proof for such improvement – none is needed. It is a reassuring safety check, though: if the results were

---

[20] Dyrland et al. 2022a.

negative it would mean that errors were made in applying the method or in the computations.

Rather than looking for some classification problem and dataset on which the decision-theoretic approach could lead to astounding improvements, we choose one where machine-learning classifiers already give excellent results, therefore difficult to improve upon; and which is characterized by a naturally high class imbalance. The classification problem is moreover of interest to us for other ongoing research projects.

The binary-classification task is a simplified version of an early-stage drug-discovery problem: to determine whether a molecule is chemically 'inactive' (class 0) or 'active' (class 1) towards one specific target protein.

Two machine-learning classifiers are considered: a Random Forest and a residual Convolutional Neural Network (ResNet), details of which are given in appendix ✎ .... The random forest takes as input a set of particular physico-chemical characteristics of a molecule, and outputs a real number in the range $[0, 1]$, corresponding to the fraction of decision trees which vote for class 1, 'active'. The convolutional-neural-network takes as input an image representing the chemical and spatial structure of the molecule, and outputs two real numbers roughly corresponding to scores for the two classes.

We use data from the ChEMBL database[21], previously used in the literature for other studies of machine-learning applications to drug discovery[22]. One set with 60% of the data is used to train and validate the two classifiers. One set with 20% is used for the calibration of the probability transducer and evaluation of the classifiers. One further data set with 20% is here used as fictive 'real data' to illustrate the results of our procedure; we call this the 'demonstration set'.

Note that *the additional demonstration dataset has an illustrative purpose only* for the sake of the present example. In a real design & evaluation of a set of candidate classifiers, the calibration set will at the same time be the evaluation test set, and no further data subset will be necessary, as explained in § 2.4.

In all data sets, class 0 ('inactive') occurs with a 91% relative frequency, and class 1 ('active') with 9%; a high class imbalance.

Technical details about the setup and training of the two classifiers and of the calculation of the probability-transducer parameters are given in appendix 🔧 ....

---

[21] Bento et al. 2014.   [22] Koutsoukas et al. 2017.

🔧 move this to technical appendix The Monte Carlo calculation of the parameters for the transducer took approximately 45 min for the random forest and 75 min for the convolutional neural network, wall-clock time (16 parallel cores each). The resulting parameters are available in our supplementary data[23].

4.2 Probability-transducer curves

Random forest

The joint probability of class c and output y, eq. (5), for the random forest is expressed by the sum

$$p(c, y) = \sum_k q_k \left[c \; \alpha_k + (1 - c) \; (1 - \alpha_k) \right] \mathrm{N}(y \mid \mu_k, \sigma_k) \qquad (13)$$

where $\mathrm{N}(\cdot)$ is a Gaussian as in eq. (9). The sum contains $2^{18} \approx 260\,000$ terms.

Figure 1 shows the probabilities of classes 1 and 0 conditional on the random-forest output: p(class 1 | output) and p(class 0 | output). It also shows the range of variability that these probabilities could have if more data were used for the calibration: with a 75% probability they would remain within the shaded regions. This variability information is provided for free by the calculation; we plan to discuss and use it more in future work.

The probabilities increase (class 1) or decrease (class 0) monotonically up to output values of around 0.9. The output, if interpreted as a probability for class 1 ('active'), tends to be too pessimistic for this class (and too optimistic for the other) in a range from roughly 0.25 to 0.95; and too optimistic outside this range. For instance, for an output of 0.3 the probability for class 1 is 40%; for an output of 1 the probability for class 1 is 92%.

Figure 2 shows the 'generative' probability densities of the random-forest output conditional on each class, p(output | class 1) and p(output | class 0). These probabilities densities are also provided for free by the parameter calculation, as explained in § 2.2. The shaded regions are again 75% possible variability intervals upon increase of the calibration dataset.

---

[23] Dyrland et al. 2022b, files `transducer_params-Random_Forest.zip` and `transducer_params-Neural_Net.zip`.
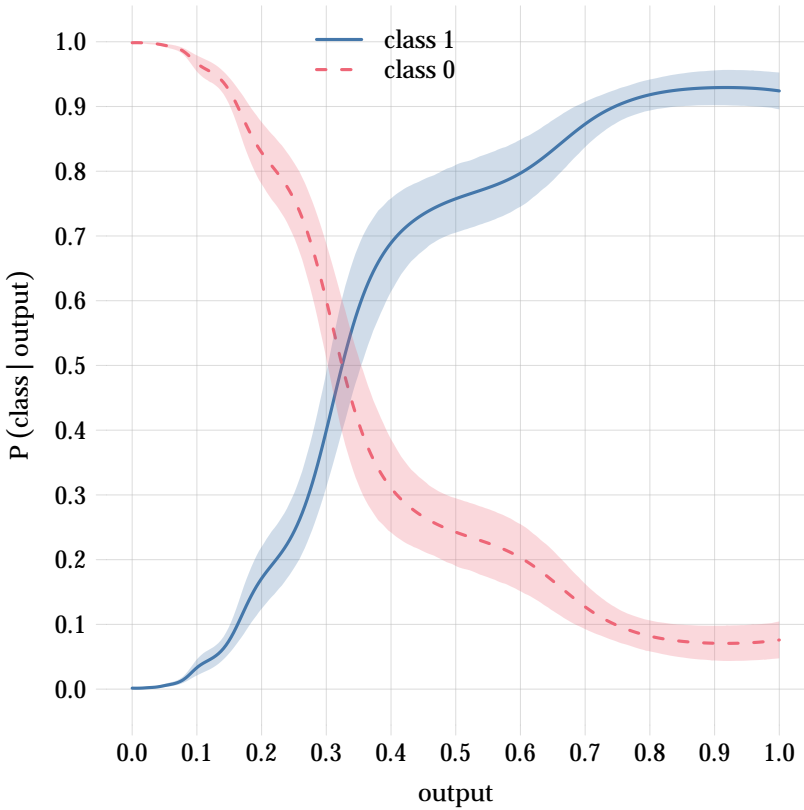
Figure 1   Probabilities of class 1 ('active', blue solid curve) and class 0 ('inactive', red dashed curve) conditional on the random-forest output. The shaded region around each curve represents its 12.5%–87.5% range of variability if more data were used to calculate the probabilities.

There is a high probability of output values close to 0 when the true class is 0 ('inactive'), and a peak density around 0.8 when the true class is 1 ('active'), as expected. The density conditional on class 0 is narrower than the one conditional on class 1 owing to the much larger proportion data in the former class. Intuitively speaking, we have seen that most data in class 1 correspond to high output values, but we have seen too few data in this class to reliably conclude, yet, that future data will show the same correspondence.
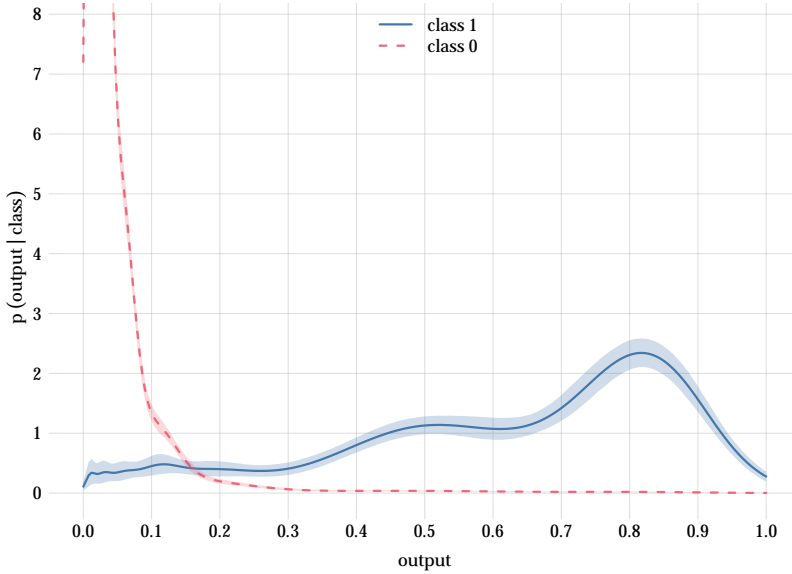
Figure 2 Probability densities of the random-forest output conditional on class 1 ('active', blue solid curve) and on class 0 ('inactive', red dashed curve, truncated). The shaded region around each curve represents its 12.5%–87.5% range of variability upon increase of the calibration dataset.

Convolutional neural network

The joint probability of class c and the bivariate output $y \equiv (y_0, y_1)$ of the convolutional neural network is expressed by the sum

$$p(c, y_0, y_1) = \sum_k q_k \left[c \, \alpha_k + (1 - c) \, (1 - \alpha_k) \right] \mathrm{N}(y_0 \mid \mu_{0k}, \sigma_{0k}) \, \mathrm{N}(y_1 \mid \mu_{1k}, \sigma_{1k}) \quad (14)$$

containing again $2^{18} \approx 260\,000$ terms, and with parameters analogous to those of eq. (13).

Figure 4.2 shows the probability of class 1 conditional on the bivariate output of the convolutional neural network, p(class 1 | outputs). It is interesting to compare this probability with the softmax function of the outputs, shown in the smaller side plot, typically used as a proxy for the probability.
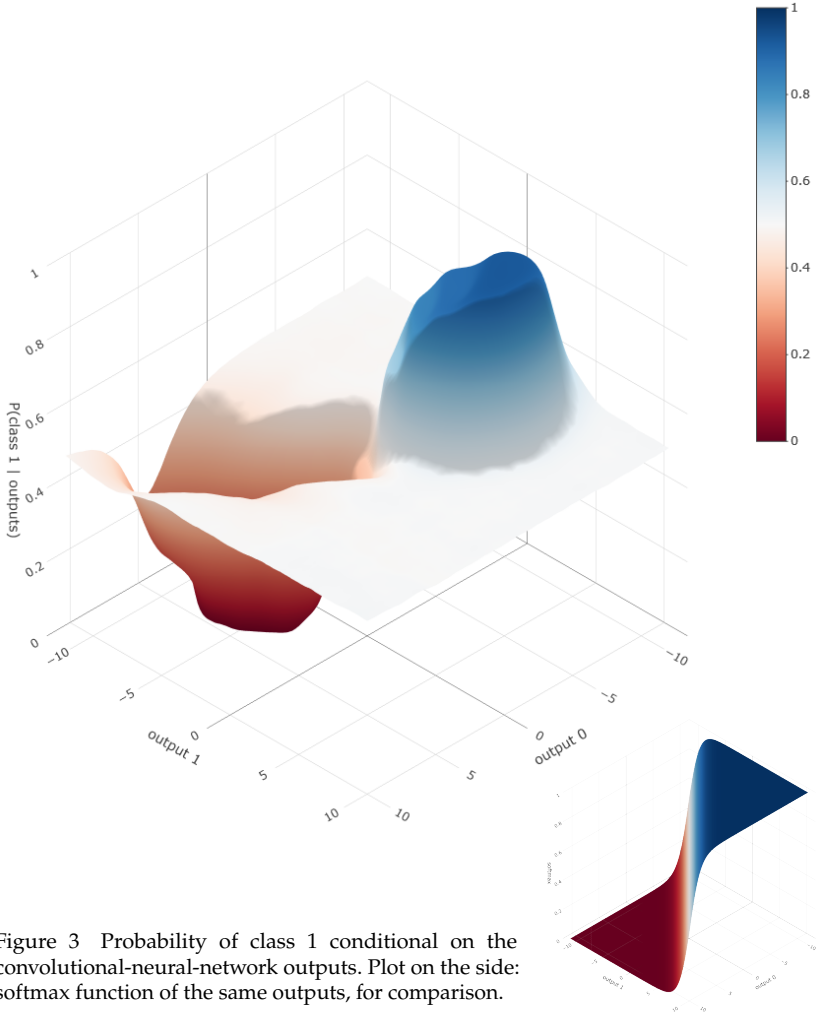
Figure 3   Probability of class 1 conditional on the convolutional-neural-network outputs. Plot on the side: softmax function of the same outputs, for comparison.

A cross-section of this probability surface along the bisector of the II and IV quadrants of the output space is shown in fig. 4, together with the cross-section of the softmax. The probability takes on extremal values, around 1% and 90%, only in very narrow ranges, and quickly returns and extrapolates to 50% everywhere else. The softmax, on the other hand, extrapolates to extreme probability values – a known problem of
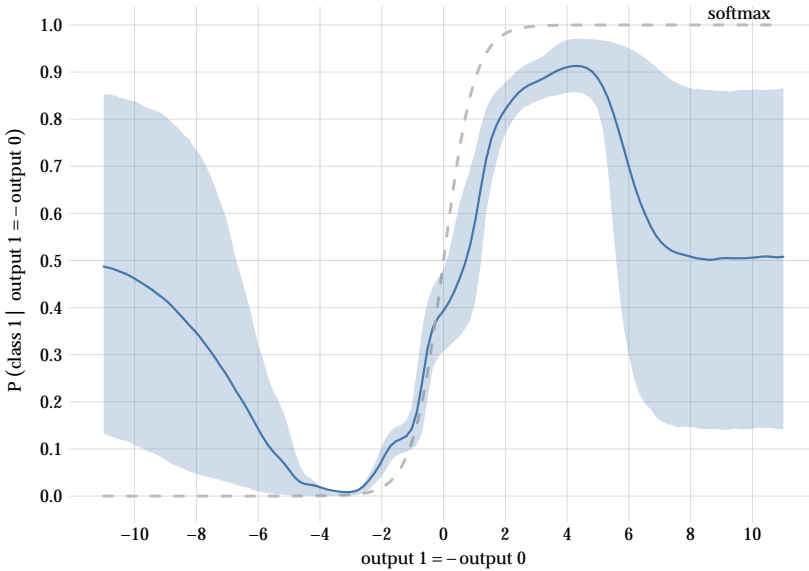
Figure 4   Cross-section of the probability surface of fig. 4.2 across the bisector of the II and IV quadrants of output space. The shaded region represents the 12.5%–87.5% range of variability upon increase of the calibration dataset. The cross-section of the softmax function (grey dashed curve) is also shown for comparison.

neural networks[24].

4.3   Evaluation

🔧 consider standard identity utility matrix, and then matrix where importance of low-frequency class is much higher.
    We assume the cost of a false positive is high (utility is very low)

$$\begin{bmatrix} 1 & 0 \\ -10 & 10 \end{bmatrix}$$

5   Sensible probabilities for classifiers

🔧 only pieces will be taken from this section

Why are probability values important? As we argue in a companion work 🔧✏️, our ultimate purpose in classification is seldom only to guess

---

[24] Gal & Ghahramani 2016.

Table 1   Results for the random-forest classifier

	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&-10\\0&10\end{bmatrix}$	$\begin{bmatrix}1&-100\\0&100\end{bmatrix}$	$\begin{bmatrix}10&0\\-10&1\end{bmatrix}$	$\begin{bmatrix}100&0\\-100&1\end{bmatrix}$
standard method	0.968	1.36	5.55	8.95	88.9
transducer & utility maximization	0.974	1.72	9.63	9.09	90.9
relative increase	+0.6%	+26%	+73%	+1.5%	+2.2%

Table 2   Results for the convolutional-neural-network classifier

	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&-10\\0&10\end{bmatrix}$	$\begin{bmatrix}1&-100\\0&100\end{bmatrix}$	$\begin{bmatrix}10&0\\-10&1\end{bmatrix}$	$\begin{bmatrix}100&0\\-100&1\end{bmatrix}$
standard method	0.959	1.52	7.24	8.63	85.6
transducer & utility maximization	0.962	1.64	9.59	9.09	90.9
relative increase	+0.3%	+8.4%	+33%	+5.4%	+6.2%

Table 3   Results for the random-forest classifier

	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&-10\\0&10\end{bmatrix}$	$\begin{bmatrix}1&-100\\0&100\end{bmatrix}$	$\begin{bmatrix}10&0\\-10&1\end{bmatrix}$	$\begin{bmatrix}100&0\\-100&1\end{bmatrix}$
standard method	0.835	3.75	34.5	3.80	33.4
transducer, generative mode	0.957	6.87	66.7	3.84	33.6
relative increase	+15%	+83%	+93%	+1.1%	+0.4%
transducer, discriminative mode	0.918	6.68	66.1	3.33	33.3

Table 4   Results for the convolutional-neural-network classifier

	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&-10\\0&10\end{bmatrix}$	$\begin{bmatrix}1&-100\\0&100\end{bmatrix}$	$\begin{bmatrix}10&0\\-10&1\end{bmatrix}$	$\begin{bmatrix}100&0\\-100&1\end{bmatrix}$
standard method	0.894	4.99	47.0	3.78	32.7
transducer, generative mode	0.930	6.80	66.7	3.74	33.4
relative increase	+4.0%	+36%	+42%	-0.9%	+0.1%
transducer, discriminative mode	0.861	6.47	66.5	3.33	33.3

a class; most often it is to choose a specific course of action, or to make a decision, among several available ones. A clinician, for example, does not simply tell a patient "you will probably not contract the disease", but has to decide among dismissal or different kinds of preventive treatment[25]. Said otherwise, in classification we must choose the *optimal* class, not the probably true one. Making optimal choices in situations of uncertainty

---

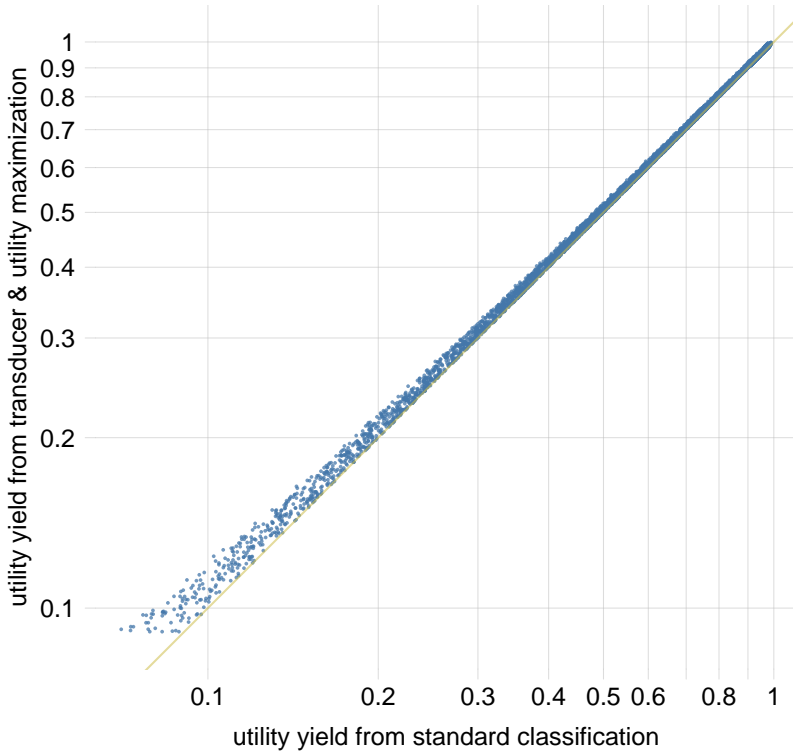[25] Sox et al. 2013; Hunink et al. 2014.

Figure 5   Utility yields obtained using the random-forest classifier in the standard way, vs those obtained using the probability transducer and utility maximization in each single classification, for a uniform distribution of possible utility matrices

is the domain of Decision Theory[26]. In order to make an optimal choice, decision theory requires the use of probability values that properly reflect our state of uncertainty.

Determining class probabilities conditional on the input features is unfortunately computationally unfeasible at present for problems that involve very high-dimensional spaces, such as image classification; in fact if an exact probabilistic analysis were possible we would not be developing machine-learning classifiers in the first place[27]. 🔧 🛠 Maybe

---

[26] Russell & Norvig 2022 ch. 15; Jeffrey 1965; North 1968; Raiffa 1970.   [27] Russell & Norvig 2022 chs 2, 12; Pearl 1988.
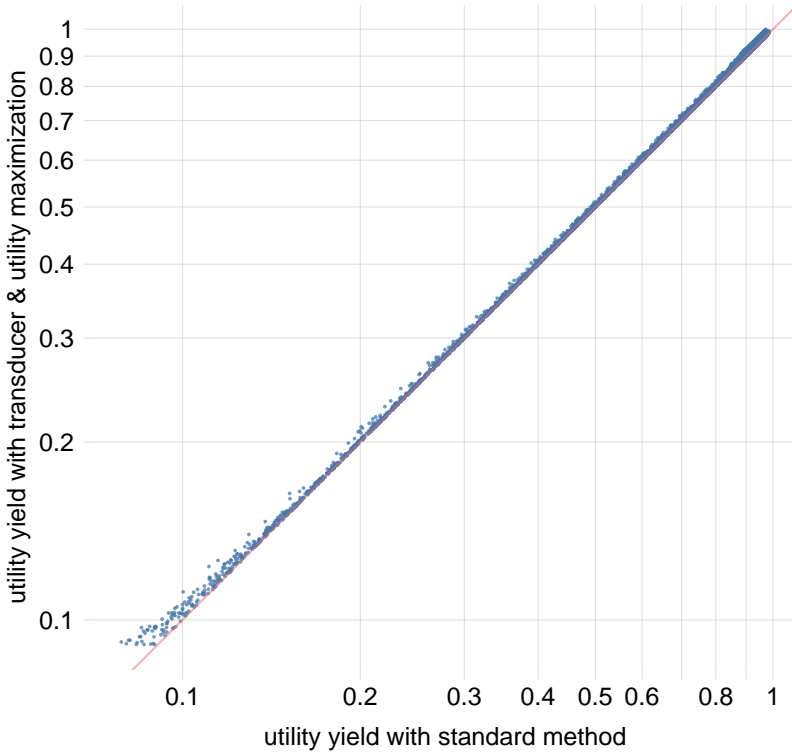
Figure 6   Utility yields obtained using the convolutional-neural-network classifier in the standard way, vs those obtained using the probability transducer and utility maximization in each single classification, for a uniform distribution of possible utility matrices

useful to add a reminder that probability theory is the *learning* theory par excellence (even if there's no 'learning' in its name)? Its rules are all about making logical updates given new data.

In the present work we propose an alternative solution that has a low computational cost and that can be applied to all commonly used classifiers, even those that only output class labels.

The essential idea comes from seeing an analogy between a classifier and a diagnostic test, such as any common diagnostic or prognostic test used in medicine for example. There are many parallels in the way machine-learning classifiers and diagnostic tests, a flu test for instance, are devised and work. Our basic motivation in using either is that
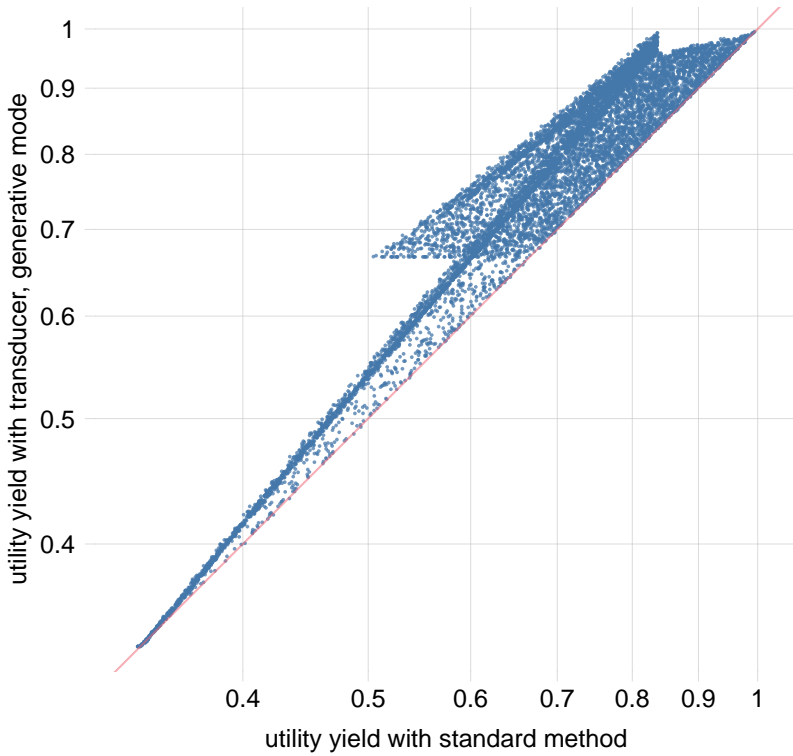
Figure 7   Utility yields obtained using the random-forest classifier in the standard way, vs those obtained using the probability transducer in generative mode, with utility maximization in each single classification, for a uniform distribution of possible utility matrices

we would like to assess some situational variable – class, pathological condition – by means of its correlation (in the general sense of the word, not the linear Pearson one; and including deterministic dependence as a particular case) with a set of 'difficult' variables that are either too complex or hidden – image pixels, presence of replicating viral agents –:

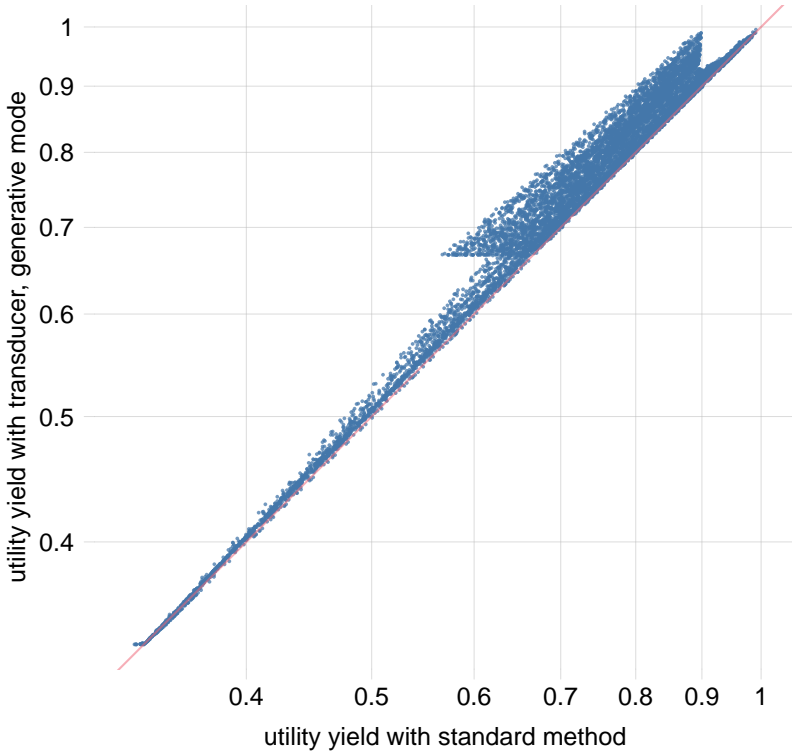$$\text{situational variable} \longleftrightarrow \text{difficult variables}$$

Figure 8 Utility yields obtained using the convolutional-neural-network classifier in the standard way, vs those obtained using the probability transducer in generative mode, with utility maximization in each single classification, for a uniform distribution of possible utility matrices

We devise an auxiliary variable – algorithm output, test result – to be correlated with the difficult variables:

$$\text{situational variable} \longleftrightarrow \text{difficult variables} \longleftrightarrow \text{aux variable}$$

We can now assess the situational variable by observing the more easily accessible auxiliary variable instead of the difficult ones. In probability language we are *marginalizing* over the difficult variables. This is the procedure dictated by the probability calculus whenever we do not have informational access to a set of variables. The correlation of the

auxiliary variable is achieved by the training process in the case of the machine-learning algorithm, and by the exploitation of biochemical processes or reactions in the case of the flu test.

The situational variable is *informationally screened* from the auxiliary variable by the difficult variables. That is, the auxiliary variable does not – in fact, cannot – contain any more information about the situational variable than that contained in the difficult variables. This means that the probability relationship between the three variables is as follows:

$$\mathrm{p}\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right) =$$

$$\sum_{\substack{\text{difficult}\\\text{variables}}} \mathrm{p}\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{difficult}\\\text{variables}\end{smallmatrix}\right) \times \mathrm{p}\left(\begin{smallmatrix}\text{difficult}\\\text{variables}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right) , \quad (15)$$

the sum running over all possible values of the difficult variables.

In the case of the diagnostic test we determine the probability $\mathrm{p}\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right)$ by carrying out the test on a representative sample of cases and collecting joint statistics between the test's output and the true situation, the presence of the flu in our example. These statistics are typically displayed in a so-called contingency table[28], akin to a confusion matrix.

Unlike the case of a diagnostic test, the output of a machine-learning classifier is usually taken at face value: if the output is a class label, that label is regarded as the true class; if the output is a unity-normalized tuple of positive numbers, that tuple is regarded as the probability distribution for the classes.

We instead propose *to treat the classifier's output just like a diagnostic test's result*. This output, discrete or continuous, is regarded as a quantity that has some correlation with the true class. This correlation can be analysed in a set of representative samples and used to calculate a sensible probability for the class given the classifier's output. This analysis only needs to be made once and is computationally cheap, because the classifier output takes values in a discrete or low-dimensional space.

This approach differs from the computationally infeasible one discussed above in that we are calculating the computationally easier probability

$$\mathrm{p}(\text{class} \mid \text{output}) \quad (16)$$

──────────────
28 Fienberg 2007; Mosteller et al. 2013.

rather than

$$p(class \mid feature) . \tag{17}$$

The former probability, as we saw in eq. (15), is the marginal

$$p(class \mid output) = \sum_{feature} p(class \mid feature) \times p(feature \mid output) . \tag{18}$$

We can thus think of this approach as a marginalization over the possible features, which is a necessary operation as have no effective access to them.

A hallmark of this approach is that we are calculating exact probabilities conditional on reduced information, rather than approximate probabilities conditional on full information. This protects us from biases that are typically present in the approximation method. The price of using reduced information is that the probabilities may be open to more variability as we collect more representative data. But as we shall see this variability is actually quite low, and moreover it can be exactly assessed.

This approach also offers the following advantages:

- It does not require any changes of the standard training procedures.

- It is easily implemented as an additional low-cost computation of a function at the end of the classifier's output, or as a replacement of a softmax-like computation.

- It does not make any assumptions such as linearity or Gaussianity.

- It yields not only the probability distribution for the classes, but also a measure of how much this distribution could change if we collected more test data (the 'probability of the probability', so to speak).

- It allows us to use the classifier both in a discriminative and generative way. That is, we can use either $p(class \mid output)$, or $p(output \mid class)$ in conjunction with Bayes's theorem. The latter approach enables us to avoid possible base-rate fallacies[29].

- It can be seamlessly integrated with a utility matrix to compute the optimal class, as shown in the companion work 🔧🖊.

In § 6 we present some notation and the general procedure for the calculation of the probabilities; more technical details are given in

---

[29] Russell & Norvig 2022 § 12.5; Axelsson 2000; Jenny et al. 2018.

appendix **??**. Section 7 explains how to augment a classifier's output with the probability calculation. Results of numerical experiments are presented in § **??**.

🔧🔧 We could show that even if we used a biased test set, the method corrects the bias (provided we know what the bias is).

6   Calculation of the probabilities: general procedure

Let us denote the class variable by C and the classifier-output variable by X. We assume that C is discrete and finite, its values can be simply renamed $1, 2, 3, \ldots$. We also assume that X is either discrete and finite (it is isomorphic to C for many classifying algorithms) or a low-dimensional tuple of real variables; a combination of both cases can also be easily accommodated. We imagine to have a sample of M such data pairs denoted collectively by D:

$$D \coloneqq \big\{ (c_1, x_1), (c_2, x_2), \ldots, (c_M, x_M) \big\} . \tag{19}$$

Let us call them *calibration data*. It must be emphasized that these are not pairs of class & feature values, but pairs of class & classifier-output values, obtained as described in § 7.

Instead of the conditional probability p(class | output), that is, $p(C \mid X)$, we can actually calculate the joint probability

$$p(C, X) \tag{20}$$

given the sample data. The computational cost is the same, but from the joint probability we can easily derive both conditionals

$$p(C \mid X) = \frac{p(C, X)}{\sum_C p(C, X)} , \tag{21}$$

$$p(X \mid C) = \frac{p(C, X)}{\sum_X p(C, X)} . \tag{22}$$

It is advantageous to have both, as we shall see in §🔧✏: if one of them is biased owing to the way the test samples were obtained, we can still use the other.

In our specific inference problem, where no time trends are assumed to exist in future data (the probability distribution for future data is exchangeable), probability theory dictates that the joint probability (20)

for a new datapoint (c_0, x_0) is equal to the *expected* frequency of that datapoint in a hypothetically infinite run of observations, that is, the average

$$p(c_0, x_0) = \int F(c_0, x_0)\, w(F)\, \mathrm{d}F \,. \tag{23}$$

This formula is the so-called de Finetti's theorem[30]. It is derived from first principles but can be intuitively interpreted: We are considering every possible long-run frequency distribution $F(\cdot\,, \cdot)$, giving it a weight $w(F)$, and then taking the weighted sum of all such distributions. The result is still a distribution, and its value at (c_0, x_0) is the probability of this datapoint.

The weight $w(F)$ – a probability density – given to a frequency distribution F is proportional to two factors:

$$w(F) \propto F(D)\, w_{\mathrm{g}}(F) \,. \tag{24}$$

- The first factor ('likelihood') $F(D)$ quantifies how well F *fits* known data of the same kind, the sample data D in our case. It is simply proportional to how frequent the known data would be, according to F:

$$F(D) = F(c_1, x_1)\, F(c_2, x_2)\, \cdots\, F(c_M, x_M) \,. \tag{25}$$

- The second factor ('prior') $w_{\mathrm{g}}(F)$ quantifies how well F *generalizes* beyond the data we have seen, owing to reasons such as physical or biological constraints for example. In our case we expect F to be somewhat smooth in X when this variable is continuous[31] 🔧🔧 add a picture – a sample from the prior over F – to illustrate the expected range of smoothness. No assumptions are made about F when X is discrete.

Formula (24) is just Bayes's theorem. Its normalization factor is the integral $\int F(D)\, w_{\mathrm{g}}(F)\, \mathrm{d}F$, which ensures that $w(F)$ is normalized.

The first factor becomes larger as the number of known data increases. Thus a large amount of data indicating a non-smooth distribution F will override any smoothness preferences embodied in the second factor. Note that no assumptions about the shape of F – no Gaussians, logistic curves, sigmoids, and so on – are made in this approach.

The integral in (23) is calculated in either of two ways, depending on whether X is discrete or continuous. For X discrete and taking on

---

[30] Bernardo & Smith 2000 ch. 4; Dawid 2013; de Finetti 1929; 1937.   [31] Cf. Good & Gaskins 1971.

the same values as the class variable C, the integral is over $\mathbf{R}^{n_c^2}$ where n_c is the number of possible classes, and can be done analytically. For X continuous, the integral is numerically approximated by a sum over a representative sample, obtained by Markov-chain Monte Carlo, of distributions F according to the weights (24). The error of this approximation can be calculated and made as small as required by increasing the number of Monte Carlo samples.

The expected value (23) is calculated for all possible values of (c_0, x_0), obtaining the full joint probability distribution $p(C, X)$. From this joint distribution we calculate the direct and inverse conditional distributions

$$p(C \mid X) = \frac{p(C, X)}{\sum_C p(C, X)} \,, \tag{26}$$

$$p(X \mid C) = \frac{p(C, X)}{\sum_X p(C, X)} \,. \tag{27}$$

It is very convenient to have both, as discussed in § 8.

The conditional distributions above are just matrices when X is discrete. For continuous X they can be regarded as n_c tuples of functions in X. We can find convenient approximate expressions, such as polynomial interpolants, for faster numerical implementations of these functions.

The integration procedure for (23) also tells us how much the probability distribution $p(C, X)$ would change if we acquired new data (a sort of 'probability of the probability').

For further mathematical details see appendix **??**.

7 Implementation in the classifier output

The implementation of our approach takes place after the training of the classifier has been carried out in the usual way. We assume that a collection of M *test data* were set aside as usual:

$$T := \big\{ (c_1, z_1), (c_2, z_2), \dots, (c_M, z_M) \big\} \,, \tag{28}$$

where the c_i are the true classes and z_i the corresponding feature values.

The M feature values z_i are given as inputs to the classifier, which produces M corresponding outputs x_i. We now consider data pairs

consisting in the true classes c_i and the outputs x_i: these are the *calibration data* discussed in § 6:

$$D := \big\{(c_1, x_1), (c_2, x_2), \ldots, (c_M, x_M)\big\} \ .$$

They are used to find the direct and inverse conditional probability distributions $p(C \mid X)$ and $p(X \mid C)$ as described in § 6.

We can finally augment our classifier either in a 'direct' or 'discriminative' way, or an 'inverse' or 'generative' way, by adding one computation step at the end of the classifier's operation:

Direct: from its output x_0 we obtain the probability for each class, $p(c \mid x_0)$.

Inverse: from its output x_0 we obtain the probability of the output itself, conditional on each class, $p(x_0 \mid c)$.

These n_c probabilities are the final output of the augmented classifier.

In the direct or discriminative case, at each new use of the classifier the output probabilities can be used together with a utility matrix to choose the *optimal* class for that case, as discussed in the companion paper🔧🔧.

In the inverse or generative case, at each new use of the classifier the probabilities for the classes are obtained via Bayes's theorem:

$$p(c) = \frac{p(x_0 \mid c) \, B(c)}{\sum_c p(x_0 \mid c) \, B(c)} \ , \tag{29}$$

where $B(c)$ is the base rate of class c. The probabilities $p(c)$ can finally be used together with a utility matrix to choose the *optimal* class.

8 Circumventing biases

9 Alternative to ensembling

🔧🔧 The idea is to implement this output-to-probability conversion in several classifiers, *in a generative way*. These probabilities can then be multiplied together and with a base rate, to obtain the 'ensembled' probabilities of the classes. This way of doing ensembling would be a rigorous application of the probability calculus; should be superior to a majority vote or similar.

10   Methods and materials

10.1   Data

The data comes from the open-access CheMBL bioactivity database[32]. The dataset used in the present work was introduced by Koutsoukas et al. (2017). The data consist in structure-activity relationships from version 20 of ChEMBL, with Carbonic Anhydrase II (ChEMBL205) as protein target.

10.2   Pre-processing

For our pre-processing pipeline, we use two different methods to represent the molecule, one for the Random Forest (RF) and one for the Convolutional Neural Network. The first method turns the molecule into a hashed bit vector of circular fingerprints called Extended Connectivity Fingerprints (ECFP)[33]. From our numerical analysis, there was little to no improvement using a 2048-bit vector over a 1024-bit vector.

For our convolutional neural network (CNN), the data is represented by converting the molecule into images of 224 pixels × 224 pixels. This is done by taking a molecule's SMILES (Simplified Molecular Input Line Entry System) string[34] from the dataset and converting it into a canonical graph structure by means of RdKit[35]. This differs from ECFP in that it represents the actual spatial and chemical structure (or something very close to it) of the molecule rather than properties generated from the molecule.

The dataset has in total 1631 active molecules and 16310 non-active molecules which act as decoys. For training, the active molecules are oversampled, as usually done with imbalanced datasets[36], to match the same number of non-active molecules.

10.3   Prediction

Virtual screening is the process of assessing chemical activity in the interaction between a compound (molecule) and a target (protein). The goal of the machine learning algorithms is to find structural features or chemical properties that show that the molecule is active towards

---

[32] Bento et al. 2014.   [33] Rogers & Hahn 2010.   [34] David et al. 2020.   [35] Landrum et al. 2017.   [36] Provost 2000.

the protein[37]. Deep neural networks have previously been shown to outperform random forests and various linear models in virtual high-throughput screening and in quantitative structure-activity relationship (QSAR) problems[38].

10.4 Chosen classifiers

The algorithms and methods used to create the models have previously been shown to give great results for a lot of different fields.

Random Forest

The first machine learning model used in the experiments is an RF model implemented in sci-kit learn[39]. RF is an ensemble of classifying or regression trees where the majority of votes is chosen as the predicted class[40]. It is known for being robust when dealing with a large number of features (as in our case), being resilient to over-fitting, and achieving good performance. And has already been shown to deliver powerful and accurate results in compound classification and QSAR analysis[41]. The following parameters were used when training the model:

Number of trees: 200

Criterion: Entropy

Max Features: Square root

Convolutional Neural Network

The second model is a pre-trained residual network (ResNet)[42] with 18 hidden layers trained on the well-known ImageNet dataset[43] by using the PyTorch framework[44]. ResNet has shown to outperform other pre-trained convolutional neural network models[45]. A ResNet with 34 hidden layers showed little to no performance gain, so we chose to go with the simpler model. The model is trained with the following hyperparameters:

---

[37] Green 2019.   [38] Koutsoukas et al. 2017.   [39] Pedregosa et al. 2011.   [40] Breiman 2001a.
[41] Svetnik et al. 2003.   [42] He et al. 2016.   [43] Russakovsky et al. 2015.   [44] Paszke et al. 2019.
[45] He et al. 2016.

Learning rate: 0.003

Optimization technique: Stochastic Gradient Descent

Activation Function: Rectified linear unit (ReLU)

Dropout: 50%

Number of epochs: 20

Loss function: Cross-entropy loss

10.5   Train, Validation, Test split

The data set is split into four parts:

- Training set: 45% of the dataset to train the model.
- Validation set: 15%, for validating the model after each epoch.
- Calibration set: 20%, for calibrating the probability transducer.
- Test set: 20%, for evaluation.

11   Summary and discussion

🔧 Add note about how (sequential) decision theory was used during World War I; see Good (1950) around § 6.2

12 An example

🔧 will delete this section

An example. Imagine you're a clinician and must attend to a patient with a particular disease. The disease may appear in two variants *I* and *II*; you are not sure which type affects your patient. For this disease there are three kinds of treatment *A*, *B*, *C* available at present, and you must choose one of them. Their efficacies, measured on some scale, depend on the disease type according to the following table:

		disease type	
		I	*II*
	A	2	−2
treatment	*B*	1	1
	C	−2	2

Treatment *A* is very effective for disease type *I* but causes harm (hence the negative value) if administered to a patient of type *II*; vice versa for treatment *C*. Treatment *B* never causes harm but is less effective on either disease type.

Which treatment do you choose?

You could say "the answer depends on the patient's disease type". This would be correct if you knew the disease type: type *I*, choose treatment *A*; type *II*, choose treatment *C*. But you do *not* know the disease type, so cannot make your answer depend on unavailable knowledge.

Rather, the answer depends *on how sure you are about the disease type*, given whatever evidence you have. This is clear in the case where you are completely uncertain about the type (and have no way to dispel your uncertainty), which could equally be either way. The rational, safest choice in this case is treatment *B*: the patient will have some relief for sure, albeit not the highest, and no harm will be done. If you are *extremely sure* that the disease type is *I* instead, then you recommend your patient to go for treatment *A*, as there's low risk for harm and greater benefits to be reaped (surgical treatments are typical examples of this case: there's always some minimal risk that a surgery goes awry, but such risk may be offset by the benefits of the more likely successful surgery). Analogously if you are sure about type *II* instead, recommending treatment *C*.

Now suppose that a shiny new machine-learning algorithm has been acquired by your clinic to help diagnosing the disease type quickly and with minimal expenses. This algorithm takes as input the results of various cheap clinical tests performed on a patient, and outputs the disease type. You use it for your patient, it outputs '*I*', so you go for treatment *A*, and the patient indeed gets better. You use it for two more patients. The outputs are *II* and *I*, so you administer treatments *C* and *A*. Both patients get better. Bliss! You use the algorithm for a third patient. Clinical tests are again made and their results fed into the algorithm. It outputs '*II*'. You therefore administer treatment *C*. But the patient is badly harmed and gets worse (and possibly sues you). To check for human error, the tests are repeated; the algorithm consistently outputs '*II*'. Eventually you perform an expensive and invasive but reliable test. It turns out the disease type is *I*, not *II*. The algorithm is simply wrong, and you chose the worst treatment.

You ask the algorithm's vendor or developers for an explanation: "isn't the algorithm supposed to tell me the actual disease type?". They tell you that no, there's always a chance that the algorithm is wrong, depending on the case. For some input values (test results) the prediction is virtually certain, but for others there may be a larger margin of error. You tell them:

> Sorry, but there has been a misunderstanding then. I don't need an algorithm that gives me a best guess, making me sometimes almost kill my patients. *I need an algorithm that tells me how sure a disease type is*. Because if there's much uncertainty I can give my patients a treatment that's guaranteed harm-free even if not the best!

This example exposes several misconceptions and issues in how classification problems and also some regression problems are often stated and faced in machine learning:

(i) The ultimate goal in a classification is often not the guess a class, but the choice among a set of alternative decisions. In a 'cat vs dog' image classification the classes are 'cat' and 'dog', and the decisions might be 'put into folder Cats' vs 'put into folder Dogs'. A clinician, however, does

not simply tell a patient "you probably have such-and-such disease", but has to decide among different kinds of treatment[46].

(ii) The alternative decisions, for example treatments in medicine, often do not correspond to the unknown classes in a one-one way; they may be more numerous than the classes.

(iii) The target is not what's most probable, but what's *optimal*. The two may be different. In the clinical example above, the clinician and patients would opt for treatment B, rather than A, even if type I had slightly higher probability than II – say 60% vs 40% (we shall show later that the threshold for this case is 70%). This is also true when the decisions have some natural one-one correspondence with the classes. Consider for example two other treatments A', B' with this efficiency table:

	disease type	
	I	*II*
A'	2	−2
B'	1	1

(treatment)

Clearly A' is best for type I, and B' for type II, but we would choose A' only if type I had quite a higher probability than II (the threshold is again 70%).

12.1 Actual utility yield

The utility matrix is not only the basis for making optimal decisions by means of expected-utility maximization. It also provides the metric to rank a set of decisions already made – for example on a test set – by some algorithm, if we know the corresponding true classes. Suppose we have N test instances, in which each class c occurs N_c times, so that $\sum_c N_c = N$. A decision algorithm made decision d when the true class was c a number M_{dc} of times. These numbers form the confusion matrix (M_{dc}) of the algorithm's output. The numbers M_{dc} are must satisfy the constraints $\sum_d M_{dc} = N_c$ for each c.

For given decision d and class c, in each of the M_{dc} instances the algorithm yielded a utility U_{dc}. The actual average utility yield in the test set is then

$$\frac{1}{N} \sum_{dc} U_{dc} M_{dc} . \tag{30}$$

---

[46] Sox et al. 2013; Hunink et al. 2014.

It is convenient to consider the average utility yield, rather than the total utility yield (without the $1/N$ factor), because if we shift the zero or change the measurement unity of our utilities then the yield changes in the same way.

The summary of decision theory just given suffices to address issues **i1**–**i4**.

In comparing, evaluating, and using machine-learning classifiers we face a number of questions and issues; some are well-known, others are rarely discussed:

i1 Choice of valuation metric. When we have to evaluate and compare different classifying algorithms or different hyperparameter values for one algorithm, we are avalanched by a choice of possible evaluation metrics: accuracy, area under curve, F_1-measure, mean square contingency[47] also known as Matthews correlation coefficient[48], precision, recall, sensitivity, specificity, and many others[49]. Only vague guidelines are usually given to face this choice. Typically one computes several of such scores and hopes that they will lead to similar ranking.

i2 Rationale and consistency. Most or all of such metrics were proposed only on intuitive grounds, from the exploration of specific problems and relying on tacit assumptions, then heedlessly applied to new problems. The Matthews correlation coefficient, for example, relies on several assumptions of Gaussianity[50], which for instance do not apply to skewed population distributions[51]. The area under the receiver-operating-characteristic curve is heavily affected by values of false-positive and false-negative frequencies, as well as by misclassification costs, that have nothing to do with those of the specific application of the classifier[52]. The F_1-measure implicitly gives correct classifications a weight that depends on their frequency or probability[53]; such dependence amounts to saying, for

[47] Yule 1912 denoted 'r' there.   [48] Matthews 1975; Fisher 1963 § 31 p. 183.   [49] Sammut & Webb 2017; see also the analysis in Goodman & Kruskal 1954; 1959; 1963; 1972. [50] Fisher 1963 § 31 p. 183 first paragraph.   [51] Jeni et al. 2013; Zhu 2020.   [52] Baker & Pinsky 2001; Lobo et al. 2008.   [53] Hand & Christen 2018.

example, "this class is rare, *therefore* its correct classification leads to high gains", which is a form of scarcity cognitive bias[54].

We are therefore led to ask: are there valuation metrics that can be proven, from first principles, to be free from biases and unnecessary assumptions?

i3 Class imbalance. If our sample data are more numerous for one class than for another – a common predicament in medical applications – we must face the 'class-imbalance problem': the classifier ends up classifying all data as belonging to the more numerous class[55], which may be an undesirable action if the misclassification of cases from the less numerous class entails high losses. 🔧 discussion and refs about cost-sensitive learning

i4 Optimality vs truth.

All the issues above are manifestly connected: they involve considerations of importance, gain, loss, and of uncertainty.

In the present work we show how issues **i1**–**i4** are all solved at once by using the principles of *Decision Theory*. Decision theory gives a logically and mathematically self-consistent procedure to catalogue all possible valuation metrics, to make optimal choices under uncertainty, and to evaluate and compare the performance of several decision algorithms. Most important, we show that implementing decision-theoretic procedures in a machine-learning classifier does not require any changes in current training practices 🧩 (possibly it may even make procedures like under- or over-sampling unnecessary!), is computationally inexpensive, and takes place downstream after the output of the classifier.

The use of decision theory requires sensible probabilities for the possible classes, which brings us to issue **??** above. In the present work we also present and use a computationally inexpensive way of calculating these probabilities from the ordinary output of a machine-learning classifier, both for classifiers such as 🧩 example here that can only output a class label, and for classifiers that can output some sort of continuous score.

🔧 Write here a summary or outlook of the rest of the paper and a summary of results:
• The admissible valuation metrics for a binary classifier form a two-dimensional family;

[54] Camerer & Kunreuther 1989; Kim & Markus 1999; Mittone & Savadori 2009. [55] Sammut & Webb 2017; Provost 2000.

that is, the choice of a specific metric corresponds to the choice of two numbers. Such choice is problem-dependent and cannot be given a priori. • Admissible metrics are only those that can be expressed as a linear function of the elements of the population-normalized confusion matrix. Metrics such as the F_1-measure or the Matthews correlation coefficient are therefore inadmissible

13 Classification from the point of view of decision theory

In using machine-learning classifiers one typically considers situations where the set of available decisions and the set of possible classes have some kind of natural correspondence and equal in number. In a 'cat vs dog' image classification, for example, the classes are 'cat' and 'dog', and the decisions could be 'put into folder Cats' vs 'put into folder Dogs'. In a medical application the classes could be 'ill' and 'healthy' and the decisions 'treat' vs 'dismiss'. In the following when we speak of 'classification' we mean a *decision* problem of this kind. The number of decisions thus equals that of classes: $n_d = n_c$.

✈ For simplicity we will focus on binary classification, $n_d = n_c = 2$, but the discussion generalizes to multi-class problems in an obvious way.

13.1 Choice of valuation metric, rationale and consistency (issues i1, i2)

13.2 Optimality vs truth (issue i4)

According to decision theory a classification algorithm should, at each application, calculate the probabilities $(p_{c|z})$ for the possible classes, given the feature z provided as input; calculate the expected utility of the available decisions according to eq. (??), using the probabilities and the utility matrix; and finally output the decision d^* having maximal expected utility:

$$d^* = \arg\max_d \sum_c U_{dc}\, p_{c|z} \ . \tag{31}$$

We assume here that the utilities are given and the same at each application – the latter assumption could be dropped, however; see the discussion in § 11.

Current common practice with algorithms capable of outputting some sort of probability-like score is simply to output the class c^* having highest probability:

$$c^* = \arg\max_c p_{c|z} \ . \tag{32}$$

As discussed in § 5, issue **i4**, this means choosing the *most probable* class, not the *optimal* class, and the two are often different, the second being what we typically want. This choice is also the one that would be made with an identity utility matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$.

How can we amend current practice for this kind of classifiers, so that they look for optimality rather than truth?

A first idea could be to simply modify the standard output step (32) into (31). It is an easily implementable and computationally cheap modification: we just multiply the probability tuple by a matrix. Such simple modification, however, has a profound implication for the training procedure: we are modifying the algorithm to output the optimal class, and therefore it should also *learn what is optimal*, not what is true: *the targets in the training and validation phases should be the optimal classes, not the true classes*. But optimality depends on the value of sensible probabilities for the specific situation of uncertainty, in this case conditional on the input features. Determining the optimal classes would thus require a probabilistic analysis that is computationally unfeasible at present for problems that involve very high-dimensional spaces, such as image classification – if an exact probabilistic analysis were possible we would not be developing machine-learning classifiers in the first place[56]. 🧩

Maybe useful to add a reminder that probability theory is the *learning* theory par excellence (even if there's no 'learning' in its name)? Its rules are all about making logical updates given new data.

Appendix: broader overview of binary classification

Let us consider our binary-classification problem from a general perspective and summarize how it would be approached and solved from first principles[57] if our computational resources had no constraints.

In our long-term task we will receive 'units' of a specific kind; the units for example could be gadgets, individuals, or investment portfolios. Each new unit will belong to one of two classes, which we can denote $X = 0$ and $X = 1$; for example they could be 'defective' vs 'non-defective', 'ill' vs 'healthy'. The class will be unknown to us. For each new unit we shall need to decide among two possible actions, which we can denote $A = \hat{0}$ and $A = \hat{1}$; for example 'discard' vs 'keep', or 'treat' vs 'dismiss'.

---

[56] Russell & Norvig 2022 chs 2, 12; Pearl 1988.   [57] Russell & Norvig 2022 part IV.

The utility of each action depends on the unknown class of the unit; we denote these utilities by $U(A \mid X)$. For each new unit we will be able to measure a 'feature' Z of a specific kind common to all units; for example Z could be a set of categorical and real quantities, or an image such as a brain scan. We have a set of units – our 'sample units' or 'sample data' – that are somehow "representative" of the units we will receive in our long-term task[58]. we know both the class and the feature of each of these sample units. Let us denote this sample information by D.

According to the principles of decision theory and probability theory, for each new unit we would proceed as follows:

1. Assign probabilities to the two possible values of the unit's class, given the value of the unit's feature $Z=z$, our sample data D, and any other available information:

$$p(X=0 \mid Z=z, D), \qquad p(X=1 \mid Z=z, D) \equiv 1 - p(X=0 \mid Z=z, D),$$
(33)

according to the rules of the probability calculus.

2. Calculate the expected utilities $\bar{U}$ of the two possible actions:

$$\bar{U}(\hat{0}) := U(\hat{0} \mid X=0)\, p(X=0 \mid Z=z, D) + U(\hat{0} \mid X=1)\, p(X=1 \mid Z=z, D)$$
$$\bar{U}(\hat{1}) := U(\hat{1} \mid X=0)\, p(X=0 \mid Z=z, D) + U(\hat{1} \mid X=1)\, p(X=1 \mid Z=z, D)$$
(34)

and choose the action having maximal expected utility.

How is the probability $p(X \mid Z=z, D)$ determined by the probability calculus? Here is a simplified, intuitive picture. First consider the case where the feature Z can only take on a small number of possible values, so that many units can in principle have the same value of Z.

Consider the collection of all units having $Z = z$ that we received in the past and will receive in the future. Among them, a proportion $F(X=0 \mid Z=z)$ belong to class 0, and a proportion $1 - F(X=0 \mid Z=z) \equiv F(X=1 \mid Z=z)$ to class 1. For example these two proportions could be 74% and 26%. Our present unit with $Z=z$ is a member of this collection. The probability $p(X=0 \mid Z=z)$ that our unit belongs to class 0, given that its feature has value z, is then intuitively equal to the proportion $F(X=0 \mid Z=z)$. Analogously for $X=1$.

---

[58] for a critical analysis of the sometimes hollow term 'representative sample' see Kruskal & Mosteller 1979a,b,c; 1980.

The problem is that we do not know the proportion $F(X{=}0 \mid Z{=}z)$. However, we expect it to be roughly equal to the analogous proportion seen in our sample data; let us denote the latter by $F_s(X{=}0 \mid Z{=}z)$:

$$F(X{=}0 \mid Z{=}z) \sim F_s(X{=}0 \mid Z{=}z) . \tag{35}$$

this is indeed what we mean by saying that our sample data are 'representative' of the future units. Later we shall discuss the case in which such representativeness is of different kinds. We expect the discrepancy between $F(X{=}0 \mid Z{=}z)$ and $F_s(X{=}0 \mid Z{=}z)$ to be smaller, the larger the number of sample data. Vice versa we expect it to be larger, the smaller the number of sample data.

If Z can take on a continuum of values, as is the case for a brain scan for example, then the collection of units having $Z{=}z$ is more difficult to imagine. In this case each unit will be unique in its feature value – no two brains are exactly alike.

🔧

old text below

Given the unit's feature Z we will assign probabilities to the possible values of the unit's class: according to the rules of the probability calculus.

As mentioned in § **??**, a decision problem under uncertainty is conceptually divided into two steps

The Suppose we have a population of units or individuals characterized by a possibly multidimensional variable Z and a binary variable $X \in \{0, 1\}$. Different joint combinations of (X, Z) values can appear in this population. Denote by $F(X{=}x, Z{=}z)$, or more simply $F(x, z)$ when there is no confusion, the number of individuals having specific joint values $(X{=}x, Z{=}z)$. This is the absolute frequency of the values (x, z). We can also count the number of individuals having a specific value of $Z{=}z$, regardless of X; this is the marginal absolute frequency $F(z)$. It is easy to see that

$$F(z) = F(X{=}0, z) + F(X{=}1, z) \equiv \sum_x F(x, z) . \tag{36}$$

Analogously for $F(x)$.

Select only the subpopulation of individuals that have a specific value $Z{=}z$. In this subpopulation, the *proportion* of individuals having

a specific value $X = x$ is $f(x \mid Z = z)$. This is the conditional relative frequency of x given that z. It is easy to see that

$$f(x \mid z) = \frac{F(x, z)}{F(z)} \, . \tag{37}$$

Now suppose that we know all these statistics about this population. An individual coming from this population is presented to us. We measure its Z and obtain the value z. What could be the value of X for this individual? We know that among all individuals having $Z = z$ (and the individual before us is one of them) a proportion $f(x \mid z)$ has $X = x$. Thus we can say that there is a probability $f(x \mid z)$ that our individual has $X = x$. And this is all we can say if we only know Z.

For this individual we must choose among two actions $\{a, b\}$. The utility of performing action a if the individual has $X = x$, and given any other known circumstances, is $U(a \mid x)$; similarly for b. If we knew the value of X, say $X = 0$, we would simply choose the action leading to maximal utility:

$$\begin{aligned}
\text{if} \quad & U(a \mid X{=}0) > U(b \mid X{=}0) \quad \text{then choose action } a, \\
\text{if} \quad & U(a \mid X{=}0) < U(b \mid X{=}0) \quad \text{then choose action } b, \\
\text{else} \quad & \text{it does not matter which action is chosen.}
\end{aligned} \tag{38}$$

But we do not know the actual value of X. We have probabilities for the possible values of X given that $Z = z$ for our individual. Since X is uncertain, the final utilities of the two actions are also uncertain; but we can calculate their *expected* values $\bar{U}(a \mid Z{=}z)$ and $\bar{U}(b \mid Z{=}z)$:

$$\begin{aligned}
\bar{U}(a \mid z) &:= U(a \mid X{=}0) \, f(X{=}0 \mid z) + U(a \mid X{=}1) \, f(X{=}1 \mid z) \, , \\
\bar{U}(b \mid z) &:= U(b \mid X{=}0) \, f(X{=}0 \mid z) + U(b \mid X{=}1) \, f(X{=}1 \mid z) \, .
\end{aligned} \tag{39}$$

Decision theory shows that the optimal action is the one having the maximal expected utility. Our choice therefore proceeds as follows:

$$\begin{aligned}
\text{if} \quad & \bar{U}(a \mid z) > \bar{U}(b \mid z) \quad \text{then choose action } a, \\
\text{if} \quad & \bar{U}(a \mid z) < \bar{U}(b \mid z) \quad \text{then choose action } b, \\
\text{else} \quad & \text{it does not matter which action is chosen.}
\end{aligned} \tag{40}$$

The decision procedure just discussed is very simple and does not need any machine-learning algorithms. It could be implemented in a

simple algorithm that takes as input the full statistics $F(X, Z)$ of the population, the utilities, and yields an output according to (40).

Our main problem is that the full statistics $F(X, Z)$ is almost universally not known. Typically we only have the statistics $F_s(X, Z)$ of a sample of individuals that come from the population of interest or from populations that are somewhat related to the one of interest. This is where probability theory steps in. It allows us to assign probabilities to all the possible statistics $F(X, Z)$. From these probabilities we can calculate the *expected* value $\bar{f}(x \mid z)$ of the conditional frequencies $f(x \mid z)$. Decision theory says that the expected value $\bar{f}(x \mid z)$ should then be used, in this uncertain case, in eq. (39) in place of the unknown $f(x \mid z)$. The decision procedure (40) can then be used again.

Probability theory says that in this particular situation the probability of a particular possible statistics $F(X, Z)$ is the product of two factors having intuitive interpretations:

- the probability of observing the statistics $F_s(X, Z)$ of our data sample, assuming the full statistics to be $F(X, Z)$. With some combinatorics it can be shown that this probability is proportional to

$$\exp\left[\sum_{X,Z} F_s(X, Z) \ln F(X, Z)\right] \tag{41}$$

  The argument of the exponential is the cross-entropy between $F_s(X, Z)$ and $F(X, Z)$; this is the reason of its appearance in the loss function used for classifiers[59].

  This factor tells us how much the possible statistics *fit* the sample data; it gives more weight to statistics with a better fit.

- the probability of the full statistics $F(X, Z)$ for reasons not present in the data, for example because of physical laws, biological plausibility, or similar.

  This factor tells us whether the possible statistics should be favourably considered, or maybe even discarded instead, for reasons that go beyond the data we have seen; in other words, whether the hypothetical statistics would *generalize* well beyond the sample data.

---

[59] Bridle 1990; MacKay 1992a.

The final probability comes from the balance between these 'fit' and 'generalization' factors. Note that the first factor becomes more important as the sample size and therefore $F_s(X, Z)$ increases; the sample data eventually determine what the most probable statistics is, if the sample is large enough.

A similar probabilistic reasoning applies if our sample data come not from the population of interest but from a population having at least the same *conditional* frequencies of as the one of interest, either $f(X \mid Z)$ or $f(Z \mid X)$. The latter case must be examined with care when our purpose is to guess X from Z. In this case we cannot use the conditional frequencies $f_s(X \mid Z)$ that appear in the data to obtain the expected value $\bar{f}(X \mid Z)$: they could be completely different from the ones of the population of interest. We must instead use the sample conditional frequencies $f_s(Z \mid X)$ to obtain the expected value $\bar{f}(Z \mid X)$, and then combine the latter with an appropriate probability $P(X)$ through Bayes's theorem:

$$\frac{\bar{f}(Z \mid X)\, P(X)}{\sum_X \bar{f}(Z \mid X)\, P(X)} \ . \tag{42}$$

The probability $P(X)$ cannot be obtained from the data, but requires a separate study or survey. In medical applications, where X represents for example the presence or absence of a disease, the probability $P(X)$ is the base rate of the disease. Direct use of $f_s(X \mid Z)$ from the data instead of (42) is the 'base-rate fallacy'[60].

In supervised learning the classifier is trained to learn the most probable $f(X \mid Z)$ from the data. The training finds the $f(X \mid Z)$ that most closely fits the conditional frequency $f_s(X \mid Z)$ of the sampled data; this roughly corresponds to maximizing the first factor (41) described above. The architecture and the parameter regularizer of the classifier play the role of the second factor.

Author contributions

The authors were so immersed in the development of the present work, that unfortunately they forgot to keep a detailed record of who did what.

---

[60] Russell & Norvig 2022 § 12.5; Axelsson 2000; Jenny et al. 2018.

Thanks

Bibliography

('de *X*' is listed under D, 'van *X*' under V, and so on, regardless of national conventions.)

Axelsson, S. (2000): *The base-rate fallacy and the difficulty of intrusion detection*. ACM Trans. Inf. Syst. Secur. **3**[3], 186–205. DOI:10.1145/357830.357849, http://www.scs.carleton.ca/~soma/id-2007w/readings/axelsson-base-rate.pdf.

Baker, S. G., Pinsky, P. F. (2001): *A proposed design and analysis for comparing digital and analog mammography special receiver operating characteristic methods for cancer screening*. J. Am. Stat. Assoc. **96**[454], 421–428. DOI:10.1198/016214501753168136.

Bar-Hillel, M. (1980): *The base-rate fallacy in probability judgments*. Acta Psychol. **44**[3], 211–233. DOI:10.1016/0001-6918(80)90046-3.

Barber, D. (2020): *Bayesian Reasoning and Machine Learning*, online update. (Cambridge University Press, Cambridge). http://www.cs.ucl.ac.uk/staff/d.barber/brml. First publ. 2007.

Bento, A. P., Gaulton, A., Hersey, A., Bellis, L. J., Chambers, J., Davies, M., Krüger, F. A., Light, Y., et al. (2014): *The ChEMBL bioactivity database: an update*. Nucleic Acids Res. **42**[D1], D1083–D1090. DOI:10.1093/nar/gkt1031. Release DOI:10.6019/CHEMBL.database.20.

Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M., West, M., eds. (2011): *Bayesian Statistics 9*. (Oxford University Press, Oxford). DOI:10.1093/acprof:oso/9780199694587.001.0001.

Bernardo, J.-M., Berger, J. O., Dawid, A. P., Smith, A. F. M., eds. (1996): *Bayesian Statistics 5*. (Oxford University Press, Oxford).

Bernardo, J.-M., Smith, A. F. (2000): *Bayesian Theory*, repr. (Wiley, New York). DOI: 10.1002/9780470316870. First publ. 1994.

Bishop, C. M. (2006): *Pattern Recognition and Machine Learning*. (Springer, New York). https://www.microsoft.com/en-us/research/people/cmbishop/prml-book.

Breiman, L. (2001a): *Random forests*. Mach. Learn. **45**[1], 5–32. DOI:10.1023/A:1010933404324.

— (2001b): *Statistical modeling: the two cultures*. Stat. Sci. **16**[3], 199–215. See also comments and rejoinder in Cox, Efron, Hoadley, Parzen, Breiman (2001).

Bridle, J. S. (1990): *Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition*. Neurocomputing **68**, 227–236. DOI: 10.1007/978-3-642-76153-9_28.

Camerer, C. F., Kunreuther, H. (1989): *Decision processes for low probability events: policy implications*. J. Policy Anal. Manag. **8**[4], 565–592. DOI:10.2307/3325045.

Cifarelli, D. M., Regazzini, E. (1979): *Considerazioni generali sull'impostazione bayesiana di problemi non parametrici. Le medie associative nel contesto del processo aleatorio di Dirichlet*. Riv. mat. sci. econ. soc. **2**[1, 2], 39–52, 95–111.

Cox, D. R., Efron, B., Hoadley, B., Parzen, E., Breiman, L. (2001): *Comments and rejoinder ['Statistical Modeling: The Two Cultures']*. Stat. Sci. **16**[216–231]. See Breiman (2001b).

Damien, P., Dellaportas, P., Polson, N. G., Stephens, D. A., eds. (2013): *Bayesian Theory and Applications*. (Oxford University Press, Oxford). DOI:10.1093/acprof:oso/9780199695607.001.0001.

David, L., Thakkar, A., Mercado, R., Engkvist, O. (2020): *Molecular representations in AI-driven drug discovery: a review and practical guide*. J. Cheminf. **12**, 56. DOI:10.1186/s13321-020-00460-5.

Dawid, A. P. (2013): *Exchangeability and its ramifications*. In: Damien, Dellaportas, Polson, Stephens (2013): ch. 2:19–29. DOI:10.1093/acprof:oso/9780199695607.003.0002.

de Finetti, B. (1929): *Funzione caratteristica di un fenomeno aleatorio*. In: *Atti del Congresso Internazionale dei Matematici:* ed. by S. Pincherle (Zanichelli, Bologna): 179–190. https://www.mathunion.org/icm/proceedings, http://www.brunodefinetti.it/Opere.htm. Transl. in Cifarelli, Regazzini (1979). See also de Finetti (1930).

— (1930): *Funzione caratteristica di un fenomeno aleatorio*. Atti Accad. Lincei: Sc. Fis. Mat. Nat. **IV**[5], 86–133. http://www.brunodefinetti.it/Opere.htm. Summary in de Finetti (1929).

— (1937): *La prévision: ses lois logiques, ses sources subjectives*. Ann. Inst. Henri Poincaré **7**[1], 1–68. http://www.numdam.org/item/AIHP_1937__7_1_1_0. Transl. in Kyburg, Smokler (1980), pp. 53–118, by Henry E. Kyburg, Jr.

Drummond, C., Holte, R. C. (2005): *Severe class imbalance: why better algorithms aren't the answer*. Eur. Conf. Mach. Learn. **2005**, 539–546. DOI:10.1007/11564096_52, https://webdocs.cs.ualberta.ca/~holte/Publications.

Dunson, D. B., Bhattacharya, A. (2011): *Nonparametric Bayes regression and classification through mixtures of product kernels*. In: Bernardo, Bayarri, Berger, Dawid, Heckerman, Smith, West (2011): 145–158. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.178.1521, DOI:10.1093/acprof:oso/9780199694587.003.0005, older version at https://www.researchgate.net/publication/228447342_Nonparametric_Bayes_Regression_and_Classification_Through_Mixtures_of_Product_Kernels.

Dunson, D. B., Pillai, N., Park, J.-H. (2007): *Bayesian density regression*. J. R. Stat. Soc. B **69**[2], 163–183.

Dyrland, K., Lundervold, A. S., Porta Mana, P. G. L. (2022a): *Does the evaluation stand up to evaluation?: A first-principle approach to the evaluation of classifiers*. Open Science Framework DOI:10.31219/osf.io/7rz8t.

— (2022b): *Bayesian augmentation of machine-learning algorithms: supplementary data*. Open Science Framework DOI:10.17605/osf.io/mfz5w.

Ferguson, T. S. (1983): *Bayesian density estimation by mixtures of normal distributions*. In: Rizvi, Rustagi, Siegmund (1983): 287–302.

Fienberg, S. E. (2007): *The Analysis of Cross-Classified Categorical Data*, 2nd ed. (Springer, New York). DOI:10.1007/978-0-387-72825-4. First publ. 1980.

Fisher, R. A. (1963): *Statistical Methods for Research Workers*, rev. 13th ed. (Hafner, New York). First publ. 1925.

— (1967): *Statistical Methods and Scientific Inference*, repr. of 2nd rev. ed. (Oliver and Boyd, Edinburgh). First publ. 1956.

Fong, E., Holmes, C. C. (2020): *On the marginal likelihood and cross-validation*. Biometrika **107**[2], 489–496. DOI:10.1093/biomet/asz077.

Gal, Y., Ghahramani, Z. (2016): *Dropout as a Bayesian approximation: representing model uncertainty in deep learning*. Proc. Mach. Learn. Res. **48**, 1050–1059. See also Appendix at arXiv DOI:10.48550/arXiv.1506.02157.

Good, I. J. (1950): *Probability and the Weighing of Evidence*. (Griffin, London).

Good, I. J., Gaskins, R. A. (1971): *Nonparametric roughness penalties for probability densities*. Biometrika **58**[2], 255–277. DOI:10.1093/biomet/58.2.255.

Goodman, L. A., Kruskal, W. H. (1954): *Measures of association for cross classifications*. J. Am. Stat. Assoc. **49**[268], 732–764. DOI:10.1080/01621459.1954.10501231. See corrections Goodman, Kruskal (1957; 1958) and also Goodman, Kruskal (1959; 1963; 1972).

— (1957): *Corrigenda: Measures of association for cross classifications*. J. Am. Stat. Assoc. **52**[280], 578. DOI:10.1080/01621459.1957.10501415. See Goodman, Kruskal (1954).

— (1958): *Corrigenda: Measures of association for cross classifications*. J. Am. Stat. Assoc. **53**[284], 1031. DOI:10.1080/01621459.1958.10501492. See Goodman, Kruskal (1954).

— (1959): *Measures of association for cross classifications. II: Further discussion and references*. J. Am. Stat. Assoc. **54**[285], 123–163. DOI:10.1080/01621459.1959.10501503. See also Goodman, Kruskal (1954; 1963; 1972).

— (1963): *Measures of association for cross classifications. III: Approximate sampling theory*. J. Am. Stat. Assoc. **58**[302], 310–364. DOI:10.1080/01621459.1963.10500850. See correction Goodman, Kruskal (1970) and also Goodman, Kruskal (1954; 1959; 1972).

— (1970): *Corrigenda: Measures of association for cross classifications. III: Approximate sampling theory*. J. Am. Stat. Assoc. **65**[330], 1011. DOI:10.1080/01621459.1970.10481142. See Goodman, Kruskal (1963).

— (1972): *Measures of association for cross classifications, IV: Simplification of asymptotic variances*. J. Am. Stat. Assoc. **67**[338], 415–421. DOI:10.1080/01621459.1972.10482401. See also Goodman, Kruskal (1954; 1959; 1963).

Green, D. V. S. (2019): *Using machine learning to inform decisions in drug discovery: an industry perspective*. In: *Machine learning in chemistry: data-driven algorithms, learning systems, and predictions*, ed. by E. O. Pyzer-Knapp, T. Laino (American Chemical Society, Washington, DC): ch. 5:81–101. DOI:10.1021/bk-2019-1326.ch005.

Gregory, P. C. (2005): *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with* Mathematica *Support*. (Cambridge University Press, Cambridge). DOI:10.1017/CBO9780511791277.

Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. A., eds. (2006): *Feature Extraction: Foundations and Applications*. (Springer, Berlin). DOI:10.1007/978-3-540-35488-8.

Hailperin, T. (2011): *Logic with a Probability Semantics: Including Solutions to Some Philosophical Problems*. (Lehigh University Press, Plymouth, UK).

Hand, D., Christen, P. (2018): *A note on using the F-measure for evaluating record linkage algorithms*. Stat. Comput. **28**[3], 539–547. DOI:10.1007/s11222-017-9746-6.

He, K., Zhang, X., Ren, S., Sun, J. (2016): *Deep residual learning for image recognition*. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) **2016**, 770–778. DOI:10.1109/CVPR.2016.90, https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Hjort, N. L. (1996): *Bayesian approaches to non- and semiparametric density estimation*. In: Bernardo, Berger, Dawid, Smith (1996): 223–253. With discussion by M. Lavine, M. Gasparini, and reply.

Hunink, M. G. M., Weinstein, M. C., Wittenberg, E., Drummond, M. F., Pliskin, J. S., Wong, J. B., Glasziou, P. P. (2014): *Decision Making in Health and Medicine: Integrating Evidence and Values*, 2nd ed. (Cambridge University Press, Cambridge). DOI:10.1017/CBO9781139506779. First publ. 2001.

Jaynes, E. T. (2003): *Probability Theory: The Logic of Science*. (Cambridge University Press, Cambridge). Ed. by G. Larry Bretthorst. First publ. 1994. DOI:10.1017/CBO9780511790423, https://archive.org/details/XQUHIUXHIQUHIQXUIHX2, http://www-biba.inrialpes.fr/Jaynes/prob.html.

Jeffrey, R. C. (1965): *The Logic of Decision*. (McGraw-Hill, New York).

Jeffreys, H. (1983): *Theory of Probability*, 3rd ed. with corrections. (Oxford University Press, London). First publ. 1939.

Jeni, L. A., Cohn, J. F., De La Torre, F. (2013): *Facing imbalanced data: recommendations for the use of performance metrics*. Proc. Int. Conf. Affect. Comput. Intell. Interact. **2013**, 245–251. DOI:10.1109/ACII.2013.47.

Jenny, M. A., Keller, N., Gigerenzer, G. (2018): *Assessing minimal medical statistical literacy using the Quick Risk Test: a prospective observational study in Germany*. BMJ Open **8**, e020847, e020847corr2. DOI:10.1136/bmjopen-2017-020847, DOI:10.1136/bmjopen-2017-020847corr2.

Kim, H., Markus, H. R. (1999): *Deviance or uniqueness, harmony or conformity? A cultural analysis*. J. Pers. Soc. Psychol. **77**[4], 785–800. DOI:10.1037/0022-3514.77.4.785.

Koutsoukas, A., Monaghan, K. J., Li, X., Huan, J. (2017): *Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data*. J. Cheminf. **9**, 42. DOI:10.1186/s13321-017-0226-y.

Kruskal, W., Mosteller, F. (1979a): *Representative sampling, I: Non-scientific literature*. Int. Stat. Rev. **47**[1], 13–24. See also Kruskal, Mosteller (1979b,c; 1980).

— (1979b): *Representative sampling, II: Scientific literature, excluding statistics*. Int. Stat. Rev. **47**[2], 111–127. See also Kruskal, Mosteller (1979a,c; 1980).

— (1979c): *Representative sampling, III: The current statistical literature*. Int. Stat. Rev. **47**[3], 245–265. See also Kruskal, Mosteller (1979a,b; 1980).

— (1980): *Representative sampling, IV: The history of the concept in statistics, 1895–1939*. Int. Stat. Rev. **48**[2], 169–195. See also Kruskal, Mosteller (1979a,b,c).

Kyburg Jr., H. E., Smokler, H. E., eds. (1980): *Studies in Subjective Probability*, 2nd ed. (Robert E. Krieger, Huntington, USA). First publ. 1964.

Landrum, G., Kelley, B., Tosco, P., sriniker, NadineSchneider, Vianello, R., gedeck, adalke, et al. (2017): *RDKit: open-source cheminformatics software*. https://www.rdkit.org. Release DOI:10.5281/zenodo.268688.

Lindley, D. V., Novick, M. R. (1981): *The role of exchangeability in inference*. Ann. Stat. **9**[1], 45–58. DOI:10.1214/aos/1176345331.

Lobo, J. M., Jiménez-Valverde, A., Real, R. (2008): *AUC: a misleading measure of the performance of predictive distribution models*. Glob. Ecol. Biogeogr. **17**[2], 145–151. DOI:10.1111/j.1466-8238.2007.00358.x, https://www2.unil.ch/biomapper/Download/Lobo-GloEcoBioGeo-2007.pdf.

MacKay, D. J. C. (1992a): *The evidence framework applied to classification networks*. Neural Comput. **4**[5], 720–736. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.5.720.

—— (1992b): *Bayesian interpolation*. Neural Comput. **4**[3], 415–447. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.3.415.

—— (1992c): *A practical Bayesian framework for backpropagation networks*. Neural Comput. **4**[3], 448–472. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.3.448.

Matthews, B. W. (1975): *Comparison of the predicted and observed secondary structure of T4 phage lysozyme*. Biochim. Biophys. Acta **405**[2], 442–451. DOI:10.1016/0005-2795(75)90109-9.

Mittone, L., Savadori, L. (2009): *The scarcity bias*. Appl. Psychol. **58**[3], 453–468. DOI:10.1111/j.1464-0597.2009.00401.x.

Mosteller, F., Fienberg, S. E., Rourke, R. E. K. (2013): *Beginning statistics with data analysis*, repr. (Dover, Mineola, USA). First publ. 1983.

Müller, P., Quintana, F. A. (2004): *Nonparametric Bayesian data analysis*. Stat. Sci. **19**[1], 95–110. http://www.mat.puc.cl/~quintana/publications/publications.html.

Murphy, K. P. (2012): *Machine Learning: A Probabilistic Perspective*. (MIT Press, Cambridge, USA). https://probml.github.io/pml-book/book0.html.

Neal, R. M., Zhang, J. (2006): *High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees*. In: Guyon, Gunn, Nikravesh, Zadeh (2006): ch. 10:265–296. DOI:10.1007/978-3-540-35488-8_11.

North, D. W. (1968): *A tutorial introduction to decision theory*. IEEE Trans. Syst. Sci. Cybern. **4**[3], 200–210. DOI:10.1109/TSSC.1968.300114, https://stat.duke.edu/~scs/Courses/STAT102/DecisionTheoryTutorial.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., et al. (2019): *PyTorch: an imperative style, high-performance deep learning library*. Adv. Neural Inf. Process. Syst. (NIPS) **32**, 8026–8037. https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library. https://pytorch.org.

Pearl, J. (1988): *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, rev. 2nd pr. (Kaufmann, San Francisco). DOI:10.1016/C2009-0-27609-4.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., et al. (2011): *Scikit-learn: machine learning in python*. J. Mach. Learn. Res. **12**[85], 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html. https://scikit-learn.org.

Porta Mana, P. G. L. (2019): *A relation between log-likelihood and cross-validation log-scores*. Open Science Framework DOI:10.31219/osf.io/k8mj3, HAL:hal-02267943, arXiv DOI:10.48550/arXiv.1908.08741.

Provost, F. (2000): *Machine learning from imbalanced data sets 101*. Tech. rep. WS-00-05-001. (AAAI, Menlo Park, USA). https://aaai.org/Library/Workshops/2000/ws00-05-001.php.

Raiffa, H. (1970): *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, 2nd pr. (Addison-Wesley, Reading, USA). First publ. 1968.

Rasmussen, C. E. (1999): *The infinite Gaussian mixture model*. Adv. Neural Inf. Process. Syst. (NIPS) **12**, 554–560. https://www.seas.harvard.edu/courses/cs281/papers/rasmussen-1999a.pdf.

Rizvi, M. H., Rustagi, J. S., Siegmund, D., eds. (1983): *Recent Advances in Statistics: Papers in Honor of Herman Chernoff on His Sixtieth Birthday*. (Academic Press, New York).

Rogers, D., Hahn, M. (2010): *Extended-connectivity fingerprints*. J. Chem. Inf. Model. **50**[5], 742–754. DOI:10.1021/ci100050t.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., et al. (2015): *ImageNet large scale visual recognition challenge*. Int. J. Comput. Vis. **115**[3], 211–252. DOI:10.1007/s11263-015-0816-y. https://www.image-net.org.

Russell, S. J., Norvig, P. (2022): *Artificial Intelligence: A Modern Approach*, Fourth Global ed. (Pearson, Harlow, UK). First publ. 1995.

Sammut, C., Webb, G. I., eds. (2017): *Encyclopedia of Machine Learning and Data Mining*, 2nd ed. (Springer, Boston). DOI:10.1007/978-1-4899-7687-1. First publ. 2011.

Self, M., Cheeseman, P. C. (1987): *Bayesian prediction for artificial intelligence*. In: *Proceedings of the third conference on uncertainty in artificial intelligence (uai'87)*, ed. by J. Lemmer, T. Levitt, L. Kanal (AUAI Press, Arlington, USA): 61–69. Repr. in arXiv DOI:10.48550/arXiv.1304.2717.

Sox, H. C., Higgins, M. C., Owens, D. K. (2013): *Medical Decision Making*, 2nd ed. (Wiley, New York). DOI:10.1002/9781118341544. First publ. 1988.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., Feuston, B. P. (2003): *Random forest: a classification and regression tool for compound classification and QSAR modeling*. J. Chem. Inf. Comput. Sci. **43**[6], 1947–1958. DOI:10.1021/ci034160g.

Thorburn, D. (1986): *A Bayesian approach to density estimation*. Biometrika **73**[1], 65–75.

Wald, A. (1949): *Statistical decision functions*. Ann. Math. Stat. **20**[2], 165–205. DOI:10.1214/aoms/1177730030.

Walker, S. G. (2013): *Bayesian nonparametrics*. In: Damien, Dellaportas, Polson, Stephens (2013): ch. 13:249–270.

Yule, G. U. (1912): *On the methods of measuring association between two attributes*. J. R. Stat. Soc. **75**[6], 579–652. DOI:10.1111/j.2397-2335.1912.tb00463.x.

Zhu, Q. (2020): *On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset*. Pattern Recognit. Lett. **136**, 71–80. DOI:10.1016/j.patrec.2020.03.030.