Open Science Framework DOI:10.31219/osf.io/***

A probability transducer and decision-theoretic augmentation 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, Dept of Radiology, Haukeland University Hospital, Bergen, Norway

Draft. 14 April 2022; updated 31 May 2022

In a classification task from a set of features, one would ideally like to have the probability of the class conditional on the features. Such probability is computationally almost impossible to find in many important cases. The primary idea of the present work is to calculate the probability of a class conditional not on the features, but on a trained classifying algorithm's output. Such probability is easily calculated and provides an output-to-probability 'transducer' that can be applied to the algorithm's future outputs. In conjunction with problem-dependent utilities, the probabilities of the transducer allows one to make the optimal choice among the classes or among a set of more general decisions, by means of expected-utility maximization. The combined procedure is a computationally cheap yet powerful 'augmentation' of the original classifier. This idea is demonstrated in a simplified drug-discovery problem with a highly imbalanced dataset. The augmentation leads to improved results, sometimes close to theoretical maximum, for any set of problem-dependent utilities. The calculation of the transducer also provides, automatically: (i) a quantification of the uncertainty about the transducer itself; (ii) the expected utility of the augmented algorithm (including its uncertainty), which can be used for algorithm selection; (iii) the possibility of using the algorithm in a 'generative mode', useful if the training dataset is biased. It is argued that the optimality, flexibility, and uncertainty assessment provided by the transducer & augmentation are dearly needed for classification problems in fields such as medicine and drug discovery.

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

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[1], the machine-learning community ought to keep in mind what the actual needs and inference contexts in such fields are. We must avoid trying (intentionally or unintentionally) to convince such fields to change their needs, or to ignore their own 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 inference problems successfully tackled by machine learning earlier on.

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:

(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

[1] Lundervold & Lundervold 2019; Chen et al. 2018; Green 2019.

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.*

(iv) The relation from image pixels to house-pet subject may be almost deterministic; so we are effectively looking for or extrapolating a function $\mathrm{pet} = f(\mathrm{pixels})$ 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 $\mathrm{disease} = f(\mathrm{scores})$ or $\mathrm{activity} = f(\mathrm{features})$ does not even exist. *We are assessing statistical relationships* $\mathrm{P}(\mathrm{disease}, \mathrm{scores})$ or $\mathrm{P}(\mathrm{activity}, \mathrm{features})$ 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[2].

Issues (i)–(iii) are simply solved by adopting the standpoint of *Decision Theory*, which we briefly review in § 3 below and discuss at length in a

[2] see also Self & Cheeseman 1987; Cheeseman 1988; 2018; Pearl 1988; MacKay 2005.

companion work[3]. 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[4].

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*[5] 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[6]; or, in terms of 'populations'[7], as the expected frequencies of the classes in the hypothetical population of units (degrees of belief and frequencies being related by de Finetti's theorem[8]). Algorithms that internally do perform probabilistic calculations, for instance naive-Bayes or logistic-regression classifiers[9], unfortunately rest on strong 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[10] 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

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

[3] Dyrland et al. 2022a. [4] cf. the analysis by Drummond & Holte 2005 (they use the term 'cost' instead of 'utility'). [5] "credibilities [that] would be agreed by all rational men if there were any rational men" Good 1966. [6] MacKay 1992a; Gal & Ghahramani 2016; Russell & Norvig 2022 chs 2, 12, 13. [7] Lindley & Novick 1981. [8] Bernardo & Smith 2000 ch. 4; Dawid 2013. [9] Murphy 2012 § 3.5, ch. 8; Bishop 2006 §§ 8.2, 4.3; Barber 2020 ch. 10, § 17.4. [10] Neal & Zhang 2006; Bishop 2006 § 5.7.

(a problem opaquely called 'density regression' or 'density estimation'[11]) 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' 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 to simple regression algorithms, does not need any changes in algorithm architecture or in training procedures, and is grounded on first principles. The probability thus obtained can be combined with utilities to perform the final classification task.

Moreover, this transducer has three other great benefits, which come automatically with its computation. First, it gives a quantification of how much the probability would change if we had further data to calculate the transducer. Second, it can give an *evaluation of the whole classifier* – including an uncertainty about such evaluation – that allows us to compare it with other classifiers and choose the optimal one. Third, 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[12], even if the algorithm was not designed for a generative use.

In § 2 we present the general idea behind the probability transducer and its calculation. Its combination with the rule of expected-utility maximization to perform classification is discussed in § 3; we call this combined use the 'augmentation' of a classifier.

In § 4 we demonstrate the implementation, use, and benefits of classifier augmentation in a concrete drug-discovery classification problem and dataset, with a random forest and a convolutional neural network classifiers.

11 Ferguson 1983; Thorburn 1986; Hjort 1996; Dunson et al. 2007. **12** Russell & Norvig 2022 § 21.2.3; Murphy 2012 § 8.6.

Section 5 offers a synopsis of further benefits and uses of the probability transducer, which are obtained almost automatically from its calculation.

Finally, we give a summary and discussion in § 6, including some teasers of further applications to be discussed in future work.

2 An output-to-probability transducer

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

Let us first consider 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 *learning* or *training data*; and

- the feature value of a *new* unit,

where the 'units' could 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 probabilities

$$\mathrm{P}(\text{class of new unit} \mid \text{feature of new unit}, \text{ classes \& features of known units}) \tag{1}$$

for each possible class.

These probabilities are obtained through the rules of the probability calculus[13]; in this case specifically through the so-called de Finetti theorem[14] which connects training data and new unit. This theorem is briefly summarized in appendix B.1.

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 $\text{class} = f(\text{feature})$, as a special case, when the probabilities are essentially 0s and 1s.

[13] Jaynes 2003; Russell & Norvig 2022 chs 12–13; Gregory 2005; Hailperin 2011; Jeffreys 1983:see further references in appendix B. [14] Bernardo & Smith 2000 ch. 4; Dawid 2013.

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[15]. Thus their output cannot be considered a probability, but *it still carries information about both class and feature variables*.

Our first step is to acknowledge 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 what we can do is to calculate the probability

$$\boxed{\mathrm{P}(\text{class of new unit} \mid \text{output for new unit})} \tag{2}$$

for each class.

Once we calculate the numerical values of these conditional probabilities, we effectively have a function that maps the algorithm's output to class probabilities. It therefore acts as an *output-to-probability transducer*.

This idea 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. A diagnostic test is useful because its result has a probabilistic relationship with the unknown of interest, say, a medical condition. This relationship is easier to quantify than the one between the condition and the more complex biological variables that the test is exploiting 'under the hood'. Likewise, the output of a classifier has a probabilistic relationship with the unknown class (owing to the training process); and this relationship is in many cases easier to quantify than the one between the class and the typically complex 'features' that are the classifier's input. 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[16]. Analogously, we need some calibration data to find the probabilities (2).

15 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. **16** Sox et al. 2013 ch. 5; Hunink et al. 2014 ch. 5; see also Jenny et al. 2018.

2.2 Calibration data

To calculate the conditional probabilities (2) 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'[17] and vice versa – exactly for the same reason why we need training data in the first place to calculate the probability of a class given the feature. Or, with a more precise term, the examples and the new unit must be *exchangeable*[18].

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. *As regards the probabilistic relation between class and output, the new unit is not exchangeable with (or 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 can be considered exchangeable with the test data. We rename such data 'transducer-calibration data', owing to its new purpose.

The probability we want to calculate is therefore

$$\mathrm{P}(\text{class of new unit} \mid \text{output for new unit},\ \text{classes \& outputs of calibr. data})\,. \tag{3}$$

For classification algorithms that output a quantity much simpler than the features, like a vector of few real components for instance, 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 probability values (4), for a fixed class and variable output, constitute a sort of 'calibration curve' (or hypersurface for multidimensional outputs) of the output-to-probability transducer for the classifier. See the concrete examples of figs 1 on page 16, and 2 on page 18. It must be

[17] for a critical analysis of the sometimes hollow term 'representative sample' see Kruskal & Mosteller 1979a,b,c; 1980. [18] Lindley & Novick 1981.

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? If the output y of the classifier is already very close to the ideal probability (1), or a monotonic function thereof, isn't the proxy probability (4) throwing it away and replacing it with something different? Quite the opposite. Owing to de Finetti's theorem, if the output y is almost identical with the ideal probability, then it becomes increasingly close to the frequency distribution of the training data, as their number increases (see appendix B.1); the same happens with the proxy probability and the frequency distribution of the calibration data. But these two data sets should be representative of each other and of future data – otherwise we would be 'learning' from irrelevant data – and therefore their frequency distributions should also converge to each other. Consequently, by transitivity we expect the proxy probability to become increasingly close to the output y. Actually, if the output is not exactly the ideal probability (1) but a monotonic function of it, the proxy probability (4) will reverse such monotonic relationship, giving us back the ideal probability.

Obviously all these considerations only hold if we have good training and calibration sets, exchangeable with (representative of) the real data that will occur in our application.

Since we are using as calibration data the data traditionally set aside as 'test data' instead, an important question arises. Do we then need a third, separate test dataset for the final evaluation and comparison of candidate classifiers or hyperparameters? This would be inconvenient: it would reduce the amount of data available for training.

The answer is no: *the calibration set automatically also acts as a test set*. In fact, from the calculations for the probability transducer, discussed in the next section, we can also arrive at a final evaluation value for the algorithm as a whole. See § 5.2 and appendix B.4 for more details about this.

It may be useful to explain why this is the case, especially for those who may mistakenly 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 the 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 (see appendix B.1). It can be shown[19] 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. It is this completeness and thoroughness that makes the ideal inference by means of the probability calculus almost computationally impossible in some cases. We thus resort to approximate but faster machine-learning methods. These methods do not typically perform such data partitions 'internally' and automatically, so we need to make them – and only approximately – by hand.

Let us stress that the performance of a classifier equipped with a probability transducer still depends on the training of the raw classifier, which is the stage where a probabilistic relation between output and class is established. If the classifier's output has no mutual information with the true class (their probabilities are essentially independent), then the transducer will simply yield a uniform probability over the classes.

The question then arises of what is the optimal division of available data into the training set and the calibration set. If the calibration set is too small, the transducer curve is unreliable. If the training set is too small, the correlation between output and class is unreliable. In future work we would like to find the optimal balance, possibly by a first-principle calculation.

2.3 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 $D \coloneqq \{c_i, y_i\}$ 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,

$$\mathrm{p}(c, y \mid D)\,, \tag{4}$$

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

[19] Porta Mana 2019; Fong & Holmes 2020; Wald 1949; many examples of this fact are scattered across the text by Jaynes 2003.

The joint probability is calculated using standard non-parametric Bayesian methods[20]. '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: figs 1 and 3 for instance show 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[21]. The end result of interest in the present section is that the probability density $p(c, y \mid D)$, with c discrete and y continuous and possibly multi-dimensional, 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. These distributions are chosen by us according to convenience; see the appendix B.1 for further details. The parameter values can be different from term to term, as indicated by the index k. The weights $\{q_k\}$ are normalized. For simplicity we shall from now on omit the dependence '$\mid \dots, D$)' on the calibration data, leaving it implicitly understood.

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:

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

[20] for introductions and reviews see e.g. Walker 2013; Müller & Quintana 2004; Hjort 1996.
[21] Dunson & Bhattacharya 2011; see also the special case discussed by Rasmussen 1999.

from which also the conditional distributions are easily obtained:

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

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

In the rest of the paper we shall use formula (7a), the probability of the class given the algorithm's output, as typically done with discriminative algorithms.

The weights and parameters $\{q_k, \alpha_k, \beta_k\}$ 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, discussed in appendix B.2. 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.

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

Now let us consider the application of the algorithm, with the probabilities calculated in the preceding section, to a new unit.

1. Fed the unit's features to the classifier, which outputs the real value y.
2. Calculate $p(c \mid y)$, for each value of c, from formula (7a), using the parameters $\{q_k, \alpha_k, \beta_k\}$ stored in memory. These are the probabilities of the classes, which are collected in a column vector (p_c).
3. The expected utility $\bar{U}_i$ of decision i is given by the matrix product of an appropriate utility matrix (U_{ic}) and the column vector (p_c):
$$\bar{U}_i \coloneqq \sum_c U_{ic}\, p_c \ . \tag{8}$$
4. Choose the decision i^* having largest $\bar{U}_i$, according to the principle of maximum expected utility:
$$\boxed{\text{choose} \quad i^* = \arg\max_i \{\bar{U}_i\} \equiv \arg\max_i \left\{ \sum_c U_{ic}\, p_c \right\}} \tag{9}$$

We call the procedure above, especially steps 2.–4., the *augmentation* of the classifier.

In step 2. we have effectively 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 (compare fig. 2).

The matrix multiplication of and subsequent selection of steps 3.–4. are computationally inexpensive; they can be considered as substitutes

[22] We apologize for the difference in notation from our companion work, where the class variable is 'j' and the utilities 'U_{ij}'

of the 'argmax' selection that typically happen at the continuous output of a classifier.

It should be noted that the utilities U_{ic} used in step 3. can either be the same for each new unit, or different from unit to unit. The augmentation procedure is therefore extremely flexible, at no additional computational cost.

4 Demonstration

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[23].

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 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.

[23] Dyrland et al. 2022a.

Two machine-learning classifiers are considered: a Random Forest and a residual Convolutional Neural Network (ResNet), details of which are given in appendix A. 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[24], previously used in the literature for other studies of machine-learning applications to drug discovery[25]. 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 § 2.2.

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 B.2.

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

$$p(c, y) = \sum_k q_k \, [c \; \alpha_k + (1 - c) \, (1 - \alpha_k)] \, \mathrm{N}(y \mid \mu_k, \sigma_k) \tag{10}$$

where $\mathrm{N}(\cdot)$ is a Gaussian as in eq. (28). The sum contains $2^{18} \approx 260\,000$ terms.

[24] Bento et al. 2014. [25] Koutsoukas et al. 2017.

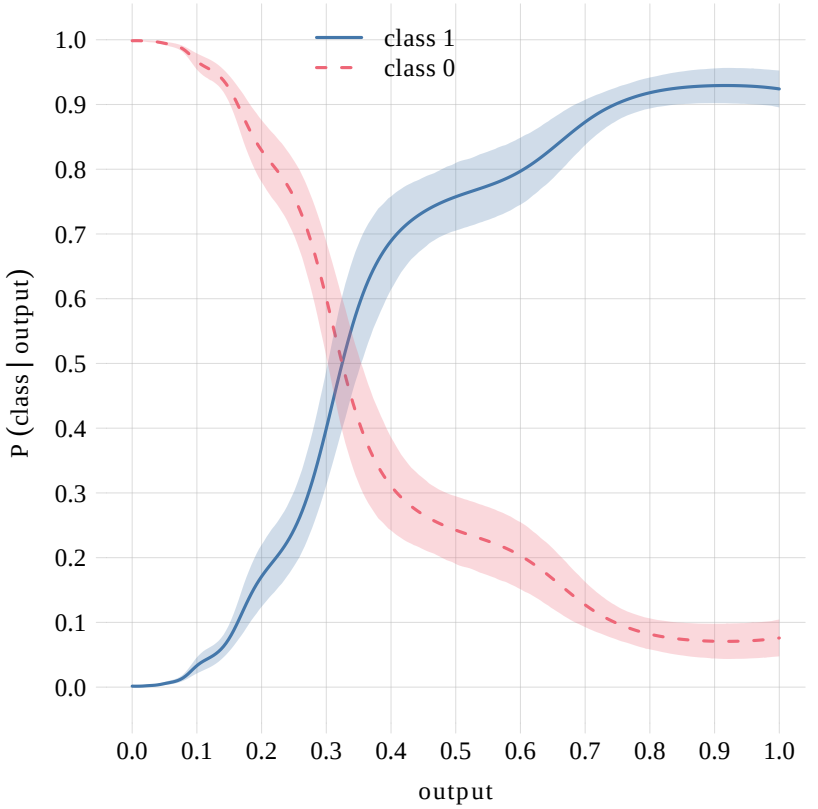

Figure 1 Probabilities of class 1 ('active', blue solid curve) and class 0 ('inactive', red dashed curve) conditional on the random-forest output. Their extremal values are 0.0014 and 0.929. The shaded region around each curve represents its 12.5%–87.5% range of possible variability if more data were used to calculate the probabilities.

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 minimum and maximum

probabilities are 0.14% and 92.9%; these values will be important for a later discussion. 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%.

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

$$p(c, y_0, y_1) = \sum_k q_k \left[c\ \alpha_k + (1-c)\,(1-\alpha_k)\right] \mathrm{N}(y_0 \mid \mu_{0k}, \sigma_{0k})\, \mathrm{N}(y_1 \mid \mu_{1k}, \sigma_{1k}) \quad (11)$$

containing again $2^{18} \approx 260\,000$ terms, and with parameters analogous to those of eq. (10).

Figure 2 shows the probability of class 1 conditional on the bivariate output of the convolutional neural network, p(class 1|outputs). Its extremal values are 0.14% and 92.3%. 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.

A cross-section of this probability surface along the bisector of the II and IV quadrants of the output space is shown in fig. 3, 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 neural networks[26]. The conservative extrapolation of the transducer is also reflected in the 75% interval of possible variability of the probability (shaded region), which becomes extremely wide at the extremities.

4.3 Results on demonstration data

The essential point of the decision-theoretic approach is that we first need to specify the utilities involved in the classification problem, because they determine (i) together with the probabilities, which class we choose in

[26] Gal & Ghahramani 2016.

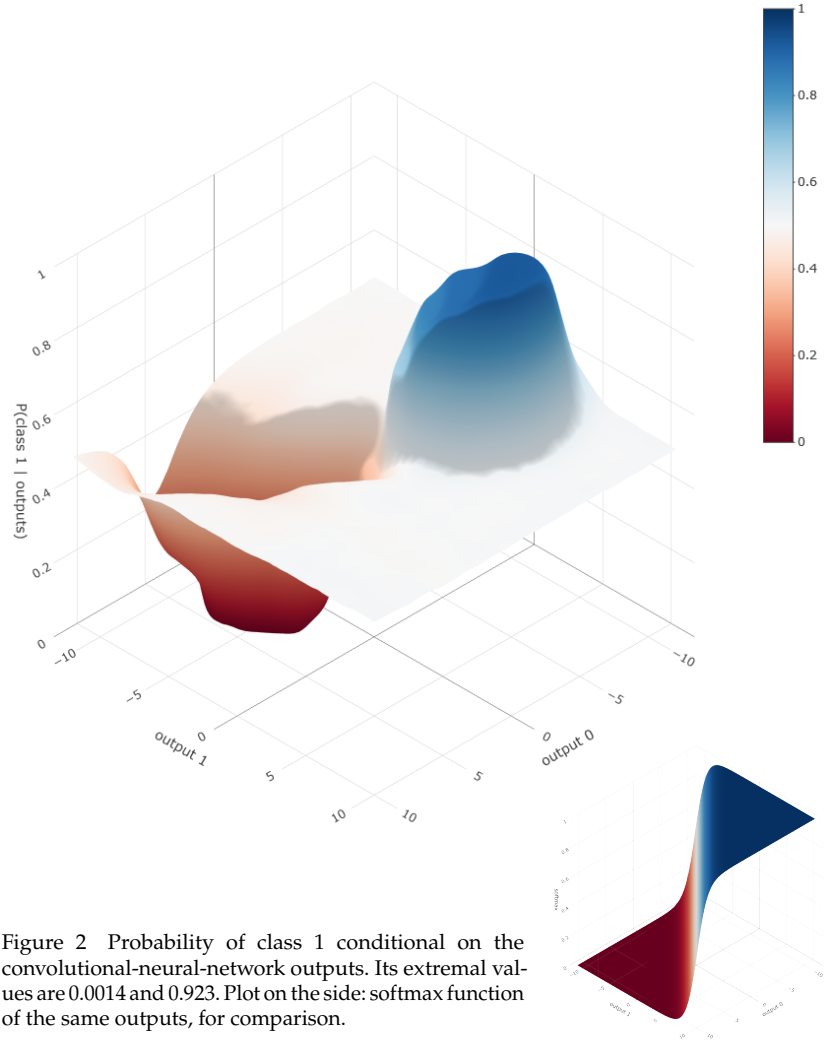

Figure 2 Probability of class 1 conditional on the convolutional-neural-network outputs. Its extremal values are 0.0014 and 0.923. Plot on the side: softmax function of the same outputs, for comparison.

each single instance; (ii) the metric to evaluate a classifier's performance. The utilities are assembled into a utility matrix which we write in the format

$$\begin{array}{c} \\ \textit{decision} \end{array} \begin{array}{c} \textit{true class} \\ \begin{array}{cc} 0 & 1 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 0 \\ 1 \end{array} \end{array} \begin{bmatrix} \text{True } 0 & \text{False } 0 \\ \text{False } 1 & \text{True } 1 \end{bmatrix} . \tag{12}$$

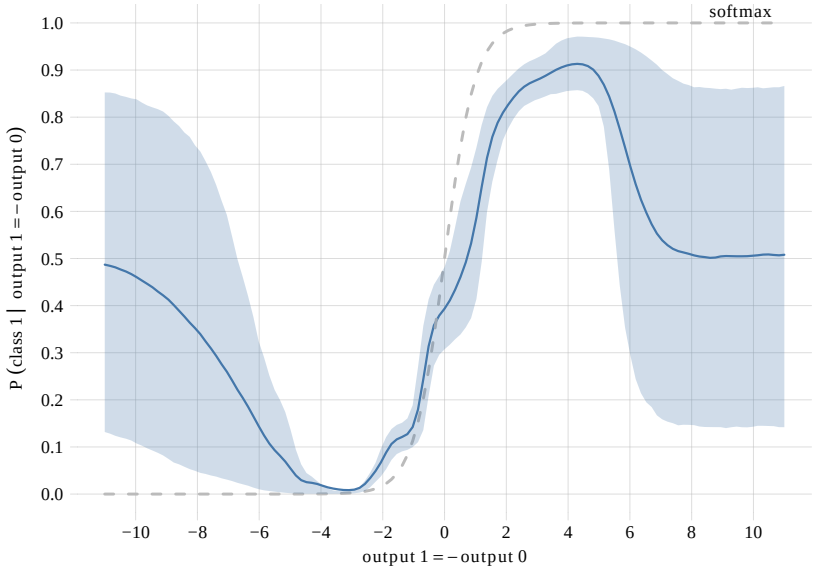

Figure 3 Cross-section of the probability surface of fig. 2 across the bisector of the II and IV quadrants of output space. The shaded region represents the 12.5%–87.5% range of possible variability upon increase of the calibration dataset. The cross-section of the softmax function (grey dashed curve) is also shown for comparison.

We call *equivalent* two utility matrices that differ by a constant additive term and a positive multiplicative term, since changes in the zero or unit of measurement of utilities do not affect comparative evaluations.

For illustration we choose four utility matrices:

$$\underset{\text{utility case I}}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} \qquad \underset{\text{utility case II}}{\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}} \qquad \underset{\text{utility case III}}{\begin{bmatrix} 1 & 0 \\ -10 & 10 \end{bmatrix}} \qquad \underset{\text{utility case IV}}{\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}} . \tag{13}$$

Case I represents any case where the correct classification of either class is equally valuable; and the incorrect classification, equally invaluable. Note that this utility matrix is equivalent to any other of the form $\left[\begin{smallmatrix} a & b \\ b & a \end{smallmatrix}\right]$ with $a > b$. Accuracy is the correct metric to evaluate this case. Case II represents any case where the correct classification of class 1, 'active' or 'positive', is ten times more valuable than that of class 0, 'inactive' or 'negative', and its incorrect classification is as damaging as correct classification is valuable. The remaining two cases are interpreted in an

analogous way. The 'value' could simply be the final average monetary revenue at the end of the drug-discovery project that typically follows any of these four situations. Of particular relevance to drug discovery, where false positives are known to be especially costly[27], is the utility matrix of case II and possibly that of case III.

We consider each of these utility matrices, in turn, to be the one underlying our classification problem. In each case we perform the classification of every item – a molecule – in the demonstration data as follows:

1. feed the features of the item to the classifier and record its output
2. feed this output to the probability transducer and record the resulting probability for class 1; form the normalized probability vector for the two classes
3. multiply the probability vector by the utility matrix to determine the expected utility of each class choice, eq. (8)
4. choose the class with higher expected utility (ties have to be decided unsystematically, to avoid biased results), eq. (9).

Confusion matrices – and a peculiar situation

Once all items in the demonstration dataset are classified, we compare their chosen classes with their true ones and compute the resulting confusion matrix, which we also write in the format (12). The confusion matrices for all cases, methods, and algorithms are presented in table 1 on page 23. Ties (both classes were equally preferable) are solved by giving half a point to each class.

The standard method produces the same confusion matrix in all four utility cases because it does not use utilities to choose a class. The augmentation produces instead a different confusion matrix in each utility case: even if the class probabilities for a give datum are the same in all cases, the threshold of acceptance varies so as to always be optimal for the utilities involved.

A peculiar case of this automatic optimization of the threshold is visible for the transducer applied to either algorithm in case IV: it leads to the confusion matrix

$$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix} \tag{14}$$

[27] Sink et al. 2010; Hingorani et al. 2019.

which means that *all* items were classified as '0', 'inactive'. How can this happen? Let us say that the probabilities for class 0 and 1, determined by the transducer from the random-forest output, are $1-p$ and p. In case IV, the expected utilities of choosing class 0 or 1 are given by the matrix multiplication $\left[\begin{smallmatrix} 10 & 0 \\ -10 & 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1-p \\ p \end{smallmatrix}\right]$:

$$\begin{aligned} &\text{choose 0: expect} \quad 10\cdot(1-p)+0\cdot p = 10-10\,p\ , \\ &\text{choose 1: expect} \quad -10\cdot(1-p)+1\cdot p = -10+11\,p\ . \end{aligned} \tag{15}$$

It is optimal to choose class 1 only if (disregarding ties)

$$-10+11\,p > 10-10\,p \qquad \text{or} \qquad p > 20/21 \approx 0.952\ , \tag{16}$$

that is, only if the probability of class 1 is higher than 95%. The threshold is so high because on the one hand there's a high cost (−10) if the class isn't actually 1, and on the other hand a high reward (10) if the class is actually 0. Now, a look at the transducer curve for the random forest, fig. 1, shows that the transducer never gives a probability higher than 93% to class 1. Similarly the transducer for the convolutional neural network, fig. 2, never reaches probabilities above 92%. So the threshold of 95% will never be met in either case, and no item will be classified as 1. It is simply never rewarding, on average, to do so[28]. It can be seen that this situation will occur with any utility matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, with $a > c$ and $d > b$, such that $\frac{a-c}{a-c+d-b} \approx 0.93$.

This peculiar situation has a notable practical consequence. We have found the transducer curve for a classifier, and see that its maximum probability for class 1 is 93%. We have assessed that the utilities involved are $\left[\begin{smallmatrix} 10 & 0 \\ -10 & 1 \end{smallmatrix}\right]$, so the threshold to classify as 1 is 95%. Then we immediately find that *there is no need to employ that classifier, in this utility case*: it is simply more profitable to automatically treat all data as class 0.

A look at the other confusion matrices of table 1 shows the effect of the automatic threshold optimization also in utility cases II and III: as the cost of some misclassification increases, the number of the correspoding misclassifications decreases.

Utility yields

We can finally assess the performance of both classifiers, with and without augmentation, on the demonstration dataset. As explained in

[28] Drummond & Holte 2005 cf. the analysis by.

our companion work[29] and summarized in § 3, the correct metric for such performance must naturally depend on the utilities that underlie the problem. It is the utility yield per datum produced by the classifier on the dataset, obtained by taking the grand sum of the products of the homologous elements of the utility matrix (U_{ij}) and the confusion matrix (C_{ij}):

$$\sum_{ij} U_{ij}\, C_{ij} \,. \tag{17}$$

The utility yields for the different cases, classifiers, and methods are presented in table 2. The maximum and minimum theoretically achievable yields, which are obtained when all data are correctly classified or incorrectly misclassified, are also shown for each case. Since the maximum and minimum differ from case to case, the table also reports the *rescaled* utilities: for each case, the rescaled utility is obtained by a change in the zero and scale of its measurement unit such that the minimum and maximum achievable yields become 0 and 1:

$$\text{rescaled utility} = \frac{\text{utility} - \text{theoretical min}}{\text{theoretical max} - \text{theoretical min}} \,. \tag{18}$$

Let us first compare the two algorithms when employed in the standard way (red). Their performance is very close to the theoretical maximum in most cases, the worse being the random forest in case II. The random forest outperforms the convolutional neural network in cases I, III, IV.

Then let us look at the performances obtained with the augmentation (blue bold). We note the following:

- The augmentation improves the performance of each algorithm in all cases. The improvement also occurs in case IV for the random forest, where the standard method already had an extremely high performance (rescaled utility of 0.988).
- In cases II and IV the augmentation improves the originally worse algorithm above the originally better one.
- In case IV the augmentation brings the utility yield to above 99% of the theoretical maximum; remember from the previous section that this is achieved by classifying all data as class 0.

[29] Dyrland et al. 2022a.

		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ -10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
Random Forest	standard method	$\begin{bmatrix} 3225 & 79.5 \\ 37 & 246.5 \end{bmatrix}$			
	augmentation	$\begin{bmatrix} 3207 & 38 \\ 55 & 288 \end{bmatrix}$	$\begin{bmatrix} 3050 & 7 \\ 212 & 319 \end{bmatrix}$	$\begin{bmatrix} 3207 & 40 \\ 55 & 286 \end{bmatrix}$	$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix}$
Neural Network	standard method	$\begin{bmatrix} 3165 & 49 \\ 97 & 277 \end{bmatrix}$			
	augmentation	$\begin{bmatrix} 3189 & 65 \\ 73 & 261 \end{bmatrix}$	$\begin{bmatrix} 2882 & 12 \\ 380 & 314 \end{bmatrix}$	$\begin{bmatrix} 3189 & 66 \\ 73 & 260 \end{bmatrix}$	$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix}$

Table 1 Confusion matrices from demonstration dataset

		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ -10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
	min achievable utility	0	−0.91	−9.09	−9.09
	max achievable utility	1	1.82	1.82	9.18
Random Forest	standard method	0.968	1.36	1.48	8.95
	augmentation	**0.974**	**1.72**	**1.54**	**9.09**
Neural Network	standard method	0.959	1.52	1.38	8.63
	augmentation	**0.962**	**1.64**	**1.41**	**9.09**
		Rescaled utility yields, eq. (18)			
Random Forest	standard method	0.968	0.834	0.969	0.988
	augmentation	**0.974**	**0.964**	**0.974**	**0.995**
Neural Network	standard method	0.959	0.890	0.960	0.970
	augmentation	**0.962**	**0.937**	**0.963**	**0.995**

Table 2 Utility yields from demonstration dataset

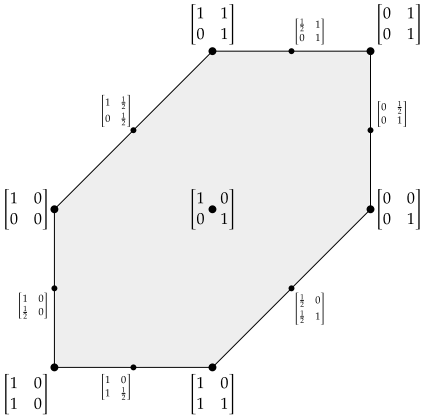

Figure 4 Space of utility matrices (modulo equivalence) for binary classification.

- With augmentation, the random forest outperforms the convolutional neural network in all cases, with a possible tie for case IV.

These were four particular cases only, though. Does the augmentation lead to an improvement (or at least to no change), on average, over all possible utility matrices? We expect this to be the case, owing to the internal consistency of decision theory.

We give evidence of this fact by considering a large number (10 000) of utility matrices selected uniformly from the utility-matrix space for binary classification. This two-dimensional space, shown in fig. 4, is discussed in our companion work[30].

For each of these utility matrices, we calculate the rescaled utility yields obtained by using either classifier in the standard way, and with the augmentation – probability-transducer & utility-based classification. The utility yields obtained in the two ways are plotted against each other in fig. 5. Histograms of their distributions are also shown on the sides.

The augmentation clearly leads to increased utility yields, especially for those cases where the standard performance of the two algorithms is particularly high or low – compare the left tails of the histograms for the standard method and augmentation. The standard method in some cases has utility yields as low as 0.76 for the random forest and 0.85 for the convolutional neural network; whereas the augmentation never leads to

30 Dyrland et al. 2022a § 3.2.

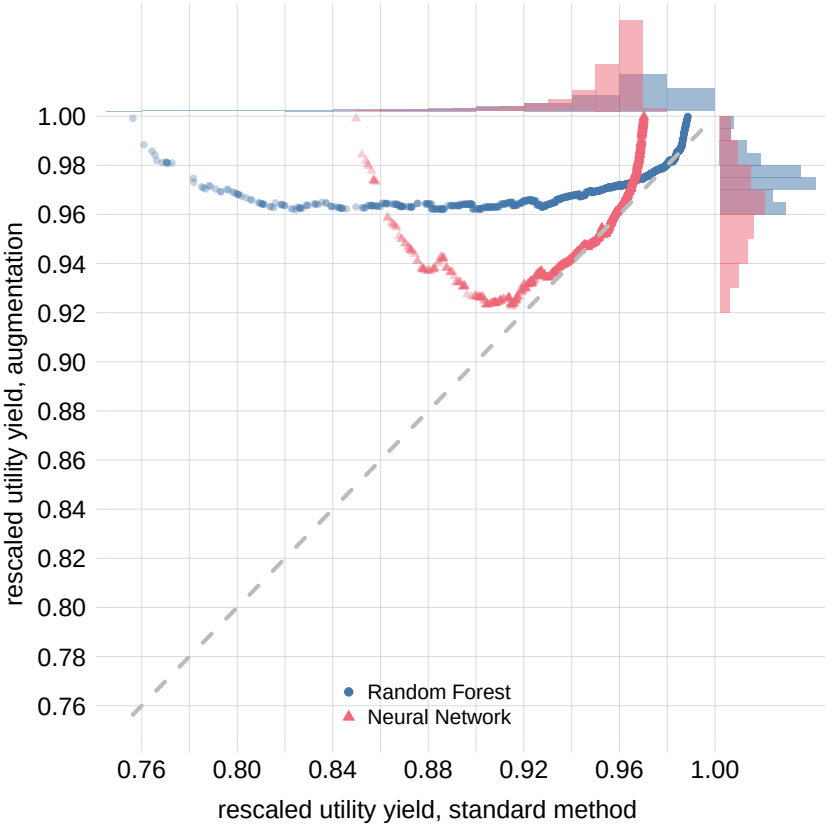

Figure 5 Rescaled utility yields obtained using the two classifiers in the standard way, vs those obtained with augmentation, for a uniform distribution of possible utility matrices over the utility-matrix space of fig. 4. The augmentation always leads to an improved utility yield, especially in cases where the standard method has a low or high performance. Owing to noise coming from numerical rounding, in some cases the yield from augmentation may appear lower than from the standard method (points below the dashed grey line).

yields below 0.96 for the random forest and 0.92 for the convolutional neural network. This explains the U-shapes of the scattered points. Note that the minimum values of the plot's axes is 0.75, so the improvement is upon utility yields that are already quite high.

There are a few apparent decreases in the utility yield, in some cases. The extremal relative decreases are −0.09% for random forest and −0.2% for convolutional neural network. Given their small magnitude,

we believe them to be caused by numerical-precision error rather than to be real decreases

4.4 From 'inactive vs active' to more general decisions

In the demonstration just discussed we assumed that the decisions available for each molecule examined were just two: 'molecule is inactive' vs 'molecule is active', corresponding to the two unknown classes. In a more general drug-discovery problem we could have a different set of decisions, for instance 'discard' vs 'promote to next examination stage' vs 'examine with different method'. Each decision would have its own utilities conditional on the two possible classes, forming a 3×2 utility matrix. The analysis and calculations of the present section would be easily generalized to such case.

5 Additional uses of the probability-transducer: an overview

The probability-transducer presented in § 2 and illustrated in the previous section has several other uses and advantages, all of which come for free or almost for free with its calculation. We give a brief overview of them in the present section, leaving a more thorough discussion and applications to future works.

The additional uses are mainly three:

- Quantification of the possible variability of the transducer probability curve.
- Evaluation of the optimal algorithm, including the uncertainty about such evaluation.
- 'Generative use' of the augmented algorithm, even if the original algorithm is not designed for generative use.

5.1 Variability of the transducer's probability curve

The output-to-probability function, eq. (4), such as those plotted in figs 1 and 2–3, is determined by the data in the calibration set. There is the question, then, of how the function could change if we used more calibration data. Such possible variability could be of importance. For example, we may find that the transducer only yields class probabilities around 0.5, and wonder whether this is just a statistical effect of a too

small calibration dataset, or whether it would persist even if we used more calibration data.

The calculation of the transducer parameters automatically tells us the probabilities of these possible variations, in the form of a set of possible alternative transducer curves, from which we can for example calculate quantiles. The shaded regions in figs 1, 7 and 3 are examples of such probability intervals. Their calculation is sketched in appendix B.3.

5.2 Expected utility of the classifying algorithm

At the end of the discussion about the calibration dataset, § 2.2, we gave our assurances that no additional data must be set apart – with a detrimental reduction in training data – for evaluation or testing purposes. This is because from the probabilities (4), obtained from the calibration data, we can also calculate *the expected, future utility yield of the augmented algorithm*, once we have specified the utility matrix underlying the particular application. More details about this calculation, which amounts to a low-dimensional integration, are given in appendix B.4; see especially formula (29).

For the random forest and convolutional neural network of the demonstration § 4, for instance, this calculation gives the expected utilities (non-rescaled) of table 3. The augmented random forest is expected to be optimal for cases I and II, possibly also in case III, although the difference in utilities is likely affected by numerical-precision error. There is no preference in case IV.

Let us emphasize again that these values are obtained *from the parameters of the transducer curve, without the need of any additional dataset*. The demonstration dataset discussed in § 4.1 was not used for their calculation. The results from that dataset, reported in table 2, corroborate these values.

One may ask: but how can you be sure that what you basically found from the calibration data will generalize to new data? The answer goes back to the discussion at the end of § 2.2, about how the probability calculus works, and to the technical details explained in appendix B: the probability calculus automatically considers all possible sets of new data that could be encountered in the future application[31].

[31] cf. Smith & Winkler 2006.

	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ -10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
Random Forest	0.973	1.68	9.59	9.08
Neural Net	0.962	1.62	9.56	9.08

Table 3 Expected utilities for the two algorithms of § 4

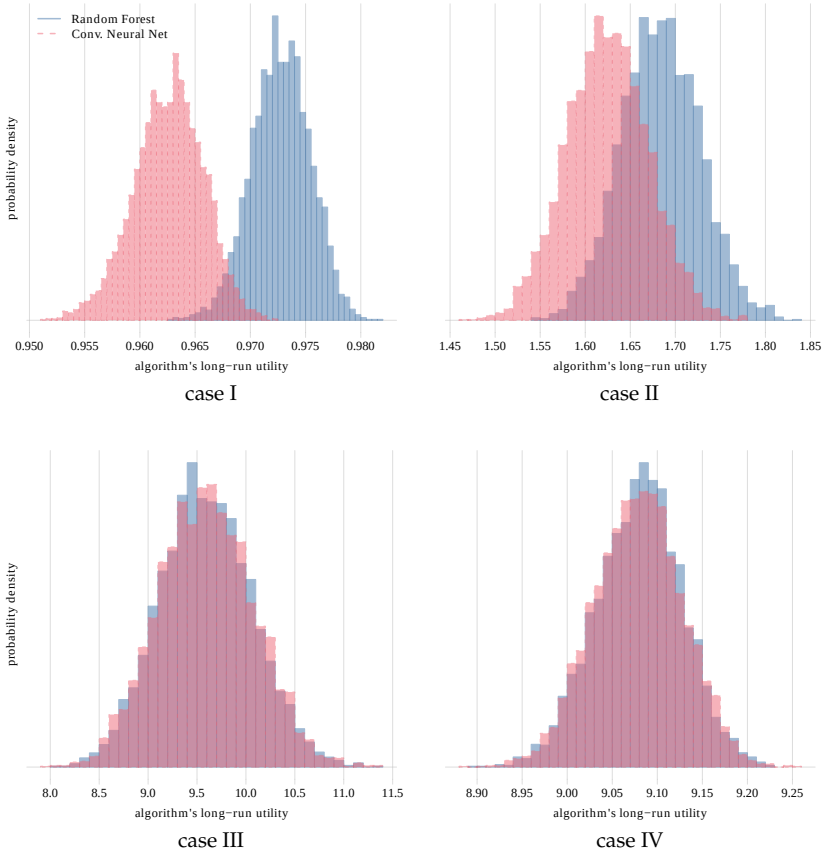

Figure 6 Probability distributions of the long-run utility yields of random forest and convolutional neural network in the four cases of § 4

In fact, the calculation of an algorithm's expected utility automatically produces a probability distribution of the possible long-run yields the algorithm could give. The distributions for the long-run utilities of the random forest and the convolutional neural network in cases I–IV of § 4 are shown in fig. 6. It can be calculated, eq. (30), that in case I the random forest will very probably, 99%, be superior to the convolutional neural network. In case II the probability is somewhat lower, 83%. In cases III and IV it is completely uncertain (50%) which algorithm will be best.

The evaluation of candidate classifiers' performances and their uncertainties are obviously extremely important for the choice and final deployment of the optimal classifier.

5.3 Discriminative and generative modes

The transducer parameters, calculated as discussed in § 2.3 and appendix B, allow us to calculate not only the 'discriminative' probability of the class given the algorithm's output, formula (7a), but also the inverse, 'generative' probability[32] of the output given the class, formula (7b). The transducer thus allow us to use the original algorithm both in 'discriminative mode' and in 'generative mode', even if it is not a generative algorithm in itself.

Having an available generative mode is extremely useful, because it is the required way to calculate the class probabilities *if the calibration and training sets do not have the same class frequencies as the real population on which the classifier will be employed*. For instance, two classes may appear in a 50%/50% proportion in the calibration set but in a 90%/10% proportion in the real population. This discrepancy in the two populations' frequencies 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; or the sampling of calibration data was poorly designed.

In such situations, the discriminative probabilities $\mathrm{p}(c \mid y)$ are usually no longer the same in the two populations either, owing to the identity

$$\mathrm{p}(c \mid y)\, \mathrm{p}(y) \equiv \mathrm{p}(y \mid c)\, \mathrm{p}(c)\,. \tag{19}$$

[32] Russell & Norvig 2022 § 21.2.3; Murphy 2012 § 8.6.

Typically, a change in $\mathrm{p}(c)$ leaves $\mathrm{p}(y \mid c)$ the same; but then both $\mathrm{p}(y)$ and $\mathrm{p}(c \mid y)$ must change as well. This means that the discriminative probabilities the algorithm and transducer have learned from the training and calibration sets are actually wrong: they cannot lead to reliable inferences on the real population.

But, as we just said, the generative probabilities $\mathrm{p}(y \mid c)$ often remain the same. And these have been automatically computed in the transducer calibration, formula (7b). We can then use them to calculate the probability of class c through Bayes's theorem, by supplying the *population prevalence* r_c of the class:

$$p(c \mid y, \text{ prevalences}) = \frac{p(y \mid c)\, r_c}{\sum_c p(y \mid c)\, r_c} \,. \tag{20}$$

The population prevalences[33], also called *base rates*[34], 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.

We recommend the outstandingly insightful discussion by Lindley & Novick 1981 on the problem of population mismatch and on which conditional probabilities to use in that case.

Figure 7 shows the 'generative' probability densities $\mathrm{p}(\text{output} \mid \text{class } 1)$, $\mathrm{p}(\text{output} \mid \text{class } 0)$ of the random-forest output from the demonstration of § 4. The shaded regions are 75% intervals of possible variability upon increase of the calibration dataset.

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.

[33] Sox et al. 2013 ch. 3; Hunink et al. 2014 § 5.1. [34] Bar-Hillel 1980; Axelsson 2000.

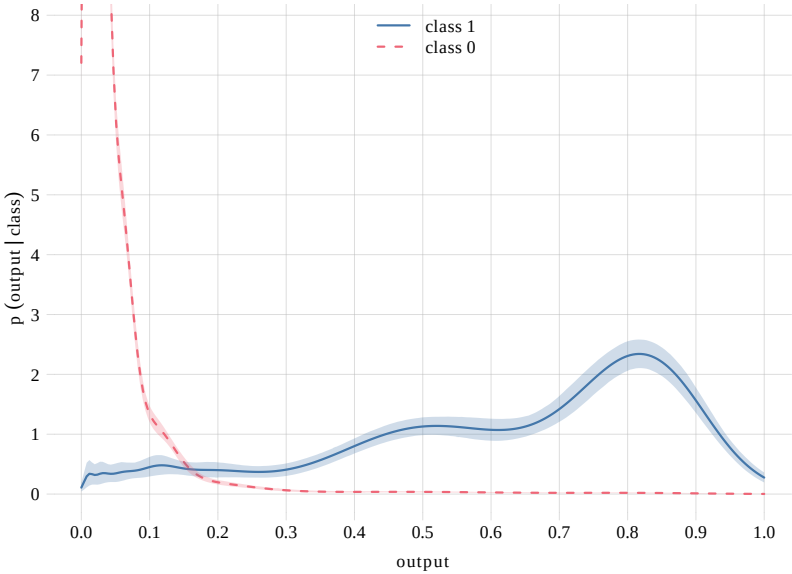

Figure 7 Probability densities of the random-forest output conditional on class 1 ('active', blue solid curve) and on class 0 ('inactive', red dashed curve, truncated). The shaded region around each curve represents its 12.5%–87.5% range of possible variability upon increase of the calibration dataset.

We can show the usefulness of using the probability transducer in generative mode by altering the class frequencies of the demonstration set: we keep all data of class 1 (the less frequent) and unsystematically select a number of data from class 0 equal to half that of class 1. This new demonstration set has thus a proportion 1/3 vs 2/3 of class 0 and class 1: their preponderance has been almost inverted. Finally we apply both classifiers in the standard way and with the augmentation in generative mode, considering again a large number of possible utility matrices, uniformly selected from their space. The rescaled utility yields are shown in fig. 8.

We see that the performance of the standard method has worsened; this is especially manifest by a comparison of the top histograms of figs 5 and 8. The median utility yield of the random forest has gone from 0.967 to 0.834; that of the convolutional neural network from 0.959 to 0.893. And yet the augmentation in generative mode is almost unaffected, the median changing from 0.974 to 0.967 for the random forest and from

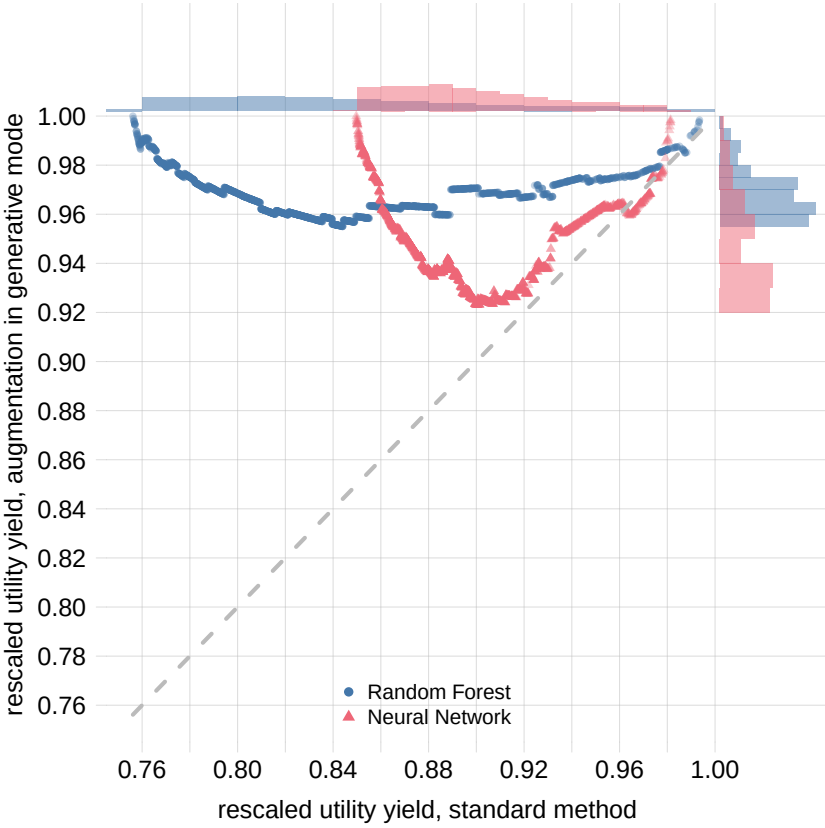

Figure 8 Rescaled utility yields obtained using the two classifiers in the standard way, vs those obtained with augmentation in generative mode, on an altered dataset with very different class balance from the training and calibration sets. The distribution of possible utility matrices is uniform over their space, as before. The utility yields of the standard method have worsened with respect to those of fig. 5, as can be seen from the histogram tails. The augmentation in generative mode, however has not suffered from this dataset mismatch.

0.961 to 0.941 for the convolutional neural network.

6 Summary and discussion

The successful application of machine-learning classifiers in fields such as medicine or drug discovery, which involve high risks and special courses of action, demands that we replace a too-simplistic view of classification

with a more articulated and flexible one. A classifier must be able to handle decisions that do not correspond to some unknown classes; it must take into account problem-specific gains and losses arising from such decisions; it must choose not what's likely, but what's optimal; and the uncertainties underlying its operation must be amenable to assessment. And it should preferably face all these requirements with methods based on first-principles guaranteeing consistency and universal applicability.

The basic theory that allows us to face most of these requirements has been around for a long time[35]: Decision Theory, whose methods keep on see-sawing in machine learning[36]. It allows us to consider decisions separate from classes, to evaluate gains and losses, and to decide what's optimal. In the present work we have tried to revive it, showing that its application is straightforward, involves little computational cost, and always leads to improvement on results obtained with standard machine-learning methods, even when these are already nearly optimal.

The main obstacle in using decision theory is that it requires proper probabilities, which in many applications might only be obtained at too high computational costs – *if these probabilities are conditional on the 'features' constituting the input to classifier*.

We have proposed the idea of *using probabilities conditional on the output of the classifier* instead. This is somehow like using the classifier in the guise of a diagnostic test, such as a typical medical test.

This probabilistic quantification is not computationally expensive, can be calculated exactly by Bayesian *model-free* (non-parametric) density-regression methods[37], and *only needs to be done once* per trained algorithm.

We have called the resulting output-to-probability function a 'probability transducer'. Concrete examples are given in fig. 1 for the output of a random-forest classifier, and in figs 2, 3 for the bivariate output of a convolutional neural network. In the latter case, the probability transducer is essentially a replacement of the popular softmax.

The quantification of the probabilistic relationship between a classifier's output and the unknown class requires a 'calibration dataset', whose role can perfectly be played by the 'test' or 'evaluation' set of standard machine-learning methodology. The calibration dataset also

[35] at least since Luce & Raiffa 1957; cf. Russell & Norvig 2022 § 1.2. [36] e.g. Self & Cheeseman 1987; Elkan 2001; Drummond & Holte 2005. [37] Dunson & Bhattacharya 2011.

delivers all necessary evaluations; thus a third, additional test set is not required.

The probability transducer gives probabilities that are easily combined with the set of utilities specific to the problem, to make a classification or a more general decision based on *maximum expected utility*, according to the principles of decision theory. This procedure is computationally inexpensive: a low-dimensional matrix multiplication followed by an 'argmax'. We have called 'augmentation' the joint use of transducer and utility-maximization. The utilities employed by the augmentation can also differ from one tested item to the other, without any changes to the computational costs.

We have demonstrated the use of the probability transducer and augmentation on a random forest and a convolutional neural network in a drug-discovery problem: classifying molecules as 'inactive' or 'active'. The problem has a naturally high class imbalance, and standard machine-learning classifiers often have nearly optimal performance on the dataset used to explore this problem. Yet, *the augmentation led to improvements for all possible choice of utilities* underlying the classification, as shown in fig. 5. The calculation of the two probability transducers' parameters took at most 75 min.

The calculation of a probability transducer from a calibration dataset also provides extremely useful additional information, for free or almost so: (a) the possible variability of the transducer function, if more calibration data were acquired; (b) the expected utility of the whole algorithm on which the transducer is used, including the uncertainty about such utility; (c) the possibility of using the classifier in a 'generative mode', giving the probability of the output conditional on the class; this is useful when the only available data for training has different statistical properties from the real-use data.

Some literature has promoted and employed the use of utilities in so called cost-sensitive learning[38]. One approach is to bake utilities into the loss function used at the training stage, so that the utilities can have an effect on the training of the classifier. This approach effectively wastes information and has computational disadvantages. First, since the optimal decision depends on the product of utilities and probabilities, the algorithm learns about this product only, and not about the two

[38] Elkan 2001; Correa Bahnsen et al. 2015; Ling & Sheng 2017.

factors separately.[39] Yet, the utilities are known, otherwise they could not be combined with the loss function. The information about them is therefore wasted. Second, if the statistics of the data involved remain the same, but the utilities suddenly change, the classifier has to be trained anew. Such a classifier cannot be used in cases where the utilities differ from one tested item to the next (see discussion above).

The method proposed in the present work does not suffer from either of these drawbacks. The training phase needs no changes, and focuses on retrieving information about the data's statistics – which is then extracted by the probability transducer. The full information contained in the utilities is used. And the utilities can even be changed on the fly during the use of the classifier.

Future directions

It is possible to construct a probability transducer that takes the output from several classifiers at once. This would be the optimal way of doing 'ensembling' from the point of view of the probability theory. In future work we plan to examine this possibility and compare it with standard ensembling methods.

As mentioned at the end of § 2.2, we also plan to assess what is the best way to split available data into the training set and the calibration set, in order to have an optimal amount of mutual information between features, class, and algorithm output (training data) and a reliable transducer (calibration data).

For the demonstration of § 4 we also tried a 'mixed' method: directly combining the output of the classifier (raw output for the random forest and standard softmax for the CNN), as it were a probability, with the utilities; and then classifying by utility maximization as usual. This method generally led to improvements with respect to the standard one, and in some cases also with respect to the probability-transducer augmentation. But on average, over the space of utility matrices, the mixed method was worse than the augmentation method, for both random forest and convolutional neural network. In future work we may try to compare the performance of the two methods with different kinds of dataset.

[39] In Elkan 2001 §§ 2–3, for example, the decision threshold of the algorithm is changed by making the algorithm learn wrong class probabilities on purpose.

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.

Thanks

KD and ASL acknowledge support from the Trond Mohn Research Foundation, grant number BFS2018TMT07, and PGLPM from The Research Council of Norway, grant number 294594.

The computations of the parameters for the probability transducer were performed on resources provided by Sigma2 – the National Infrastructure for High Performance Computing and Data Storage in Norway (project NN8050K).

KD would like to thank family for endless support; partner Synne for constant love, support, and encouragement; and the developers and maintainers of Python, FastAi, PyTorch, scikit-learn, NumPy and RDKit for free open source software and for making the experiments possible.

PGLPM thanks Maja, Mari, Miri, Emma for continuous encouragement and affection; Buster Keaton and Saitama for filling life with awe and inspiration; and the developers and maintainers of LaTeX, Emacs, AUCTeX, Open Science Framework, R, Nimble, Inkscape, LibreOffice, Sci-Hub for making a free and impartial scientific exchange possible.

Appendices: mathematical and technical details

A Algorithms and data used in the demonstration

A.1 Data

The data comes from the open-access ChEMBL bioactivity database[40]. 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.

A.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 (CNN). The first method turns the molecule into a hashed bit vector of circular fingerprints called Extended Connectivity Fingerprints (ECFP)[41]. 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, 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[42] from the dataset and converting it into a canonical graph structure by means of RdKit[43]. 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[44], to match the same number of non-active molecules.

A.3 Prediction

Virtual screening is the process of assessing chemical activity in the interaction between a compound (molecule) and a target (protein). The

[40] Bento et al. 2014. [41] Rogers & Hahn 2010. [42] David et al. 2020. [43] Landrum et al. 2017. [44] Provost 2000.

goal of the machine learning algorithms is to find structural features or chemical properties that show that the molecule is active towards the protein[45]. 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[46].

A.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[47]. RF is an ensemble of classifying or regression trees where the majority of votes is chosen as the predicted class[48]. 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[49]. 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)[50] with 18 hidden layers trained on the well-known ImageNet dataset[51] by using the PyTorch framework[52]. ResNet has shown to outperform other pre-trained convolutional neural network models[53]. A ResNet with 34 hidden layers

[45] Green 2019. [46] Koutsoukas et al. 2017. [47] Pedregosa et al. 2011. [48] Breiman 2001. [49] Svetnik et al. 2003. [50] He et al. 2016. [51] Russakovsky et al. 2015. [52] Paszke et al. 2019. [53] He et al. 2016.

showed little to no performance gain, so we chose to go with the simpler model. The model is trained with the following hyperparameters:

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

A.5 Dateset 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.
- Demonstration set: 20%, for evaluation.

B Mathematical details and computation of the transducer

The notation is the one used in § 2.3: the class is denoted c and the algorithm output y. In our demonstration c takes on values in $\{0, 1\}$, and y either in $[0, 1]$ or in $\mathbf{R}^2$; but the method can be applied to more general cases, such as continuous but low-dimensional spaces for both c and y, or combinations of continuous and discrete spaces. For convenience we use a single symbol for the pair $d \coloneqq (c, y)$.

For general references about the probability calculus and concepts and specific probability distributions see Jaynes 2003; MacKay 2005; Jeffreys 1983; Gregory 2005; Bernardo & Smith 2000; Hailperin 1996; Good 1950; Johnson et al. 1996; 2005; 1994; 1995; Kotz et al. 2000.

B.1 Exchangeability and expression for the probability transducer

There is a fundamental theorem in the probability calculus that tells us how extrapolation from known units – molecules, patients, widgets, images – to new, unknown units takes place: de Finetti's theorem[54]. It is a consequence of the assumption that our uncertainty is invariant or 'exchangeable' under permutations of the labelling or order of the units (therefore it does not apply to time series, for example).

De Finetti's theorem states that the probability density of a value d_0 for a new unit '0', conditional on data D, is given by the following integral:

$$\mathrm{p}(d_0 \mid D) = \int F(d_0)\, w(F \mid D)\, \mathrm{d}F \,, \tag{21}$$

which can be given an intuitive interpretation. We consider every possible long-run frequency distribution $F(d)$ of data; give it a weight density $w(F|D)$ which depends on the observed data; and then take the weighted sum of all such long-run frequency distributions.

The weight $w(F|D)$ given to a frequency distribution F is proportional to two factors:

$$w(F \mid D) \propto F(D)\, w_{\mathrm{g}}(F) \,. \tag{22}$$

- The first factor ('likelihood') $F(D)$ quantifies how well F *fits* known data of the same kind, in our case the calibration data $D \coloneqq \{d_1, \dots, d_M\}$. It is simply proportional to how frequent the known data would be, according to F:

$$F(D) \coloneqq F(d_1) \cdot F(d_2) \cdot \dots \cdot F(d_M) \equiv \exp\left[M \sum_d \hat{F}(d) \ln F(d)\right] , \tag{23}$$

  where $\hat{F}(d)$ is the frequency distribution observed in the data.

- 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[55]. No assumptions are made about F when X is discrete.

Formula (22) 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.

[54] Bernardo & Smith 2000 ch. 4; Dawid 2013; de Finetti 1929; 1937. [55] Cf. Good & Gaskins 1971.

The exponential expression in eq. (23) is proportional to the number M of data, and in it we recognize the cross-entropy between the observed frequency distribution $\hat{F}$ and F. This has two consequences. First, it makes the final probability $p(d_0)$ increasingly identical with the distribution $\hat{F}(d)$ observed in the data, because the average (21) gets more and more concentrated around $\hat{F}$. Second, 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 – Gaussians, logistic curves, sigmoids, or similar – are made in this approach.

The integral in (21) is calculated in either of two ways, depending on whether d is discrete or continuous. For d discrete, the integral is over $\mathbf{R}^n$, where n is the number of possible values of d, and can be done analytically. For d with continuous components, the integral is numerically approximated by a sum over T representative samples, obtained by Markov-chain Monte Carlo, of distributions F according to the weights (22):

$$\mathrm{p}(d_0 \mid D) = \int F(d_0)\, w(F \mid D)\, \mathrm{d}F \approx \frac{1}{T} \sum_{t=1}^{T} F_t(d_0)\,. \tag{24}$$

The error of this approximation can be calculated and made as small as required by increasing the number of Monte Carlo samples.

We must find a way to express any kind of distribution $F(d)$. As mentioned in § 2.3, this is done by writing it as

$$F(c, y) = \sum_l w_l\, A(c \mid \alpha_l)\, B(y \mid \beta_l)\,, \tag{25}$$

where the sum has a large number of terms[56], $\{w_l\}$ are normalized weights, and $A(c \mid \alpha)$, $B(y \mid \beta)$ are distributions, possibly the product of further one-dimensional distributions. Effectively we are expressing $F(\,\cdot\,)$ by the 'coordinates' (w_l, α_l, β_l) in a space of extremely high dimensions.

This representation[57] has several advantages:

- Its marginal distributions for c and y are also of the form (25), as shown in § 2.3, and easily computable.

[56] see Ishwaran & Zarepour 2002 on why the number of terms does not need to be infinite.
[57] promoted by Dunson & Bhattacharya 2011; see also Rasmussen 1999.

- Its conditional distributions for c given y and vice versa have also a form similar to eq. (25) and easily computable.
- It can be used with conjugate priors.
- The final probability $p(c, y)$, approximated by the sum (24), has also the form (25):

$$\mathrm{p}(c_0, y_0) \approx \sum_{t,l} \frac{w_{t,l}}{T}\, A(c_0 \mid \alpha_{t,l})\, B(y_0 \mid \beta_{t,l})\,. \tag{26}$$

  This is the expression given in § 2.3 with k running over both indexes t, l and with $q_k \coloneqq w_{t,l}/T$.

In our demonstration of § 4, where the class variable c takes on conventional values $\{0, 1\}$, we use a Bernoulli distribution for the first:

$$A(c \mid \alpha) = c\ \alpha + (1-c)\,(1-\alpha) \equiv \begin{cases} \alpha & \text{if } c = 1, \\ 1-\alpha & \text{if } c = 0\ ; \end{cases} \tag{27}$$

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

$$B(y \mid \beta) = \mathrm{N}(y \mid \mu, \sigma) \coloneqq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right], \tag{28}$$

or a product of such Gaussians, each with its own parameters, if y is multidimensional. For the random-forest output $y \in [0, 1]$ such a Gaussian should in principle be truncated; we did not use any truncation as the error committed is small and the computation much faster.

B.2 Markov-chain Monte Carlo sampling

The samples $\{F_t(\,\cdot\,)\}$ of the sum (24) – these are samples of the distribution $w(F)$ – are obtained through Markov-chain Monte Carlo; specifically Gibbs sampling[58]. Effectively we obtain samples of the coordinates (w_l, α_l, β_l), and the prior $w_{\mathrm{g}}(F)$ is a prior over these coordinates.

For the demonstration of § 4 we use a Dirichlet distribution for (w_l), a beta distribution for (α_l), a Gaussian distribution for (μ_l), and a gamma distribution for $(1/{\sigma_l}^2)$.

[58] Neal 1993; MacKay 2005 ch. 29.

The Markov-chain Monte Carlo sampling scheme is implemented using the R[59] package nimble[60], and uses 16 parallel chains[61]. The sampling took approximately 45 min for the random forest and 75 min for the convolutional neural network, wall-clock time. The resulting parameters are available in our supplementary data[62].

B.3 Assessment of the possible variability of the probability

In §🔧 we mentioned that the calculation of the transducer parameters automatically also tells us how much the probabilities curves could change if we used more data for the calibration. The range of this possible variability was shown for example in figs 1, 7, and 3 of the demonstration.

This possible variability is encoded in the weight $w(F \mid D)$, which can be interpreted as the probability distribution of the long-run frequency distribution F; remember that the probability $\mathrm{p}(d_0 \mid D)$, for the new unit, eq. (21), becomes closer and closer to the distribution F at which $w(F \mid D)$ peaks, as the number of data increases.

In the approximation (24), the probability $w(F \mid D)$ is effectively represented by a large number of samples $\{F_t\}$ from it. Plotting these samples alongside $\mathrm{p}(d_0 \mid D)$ gives an approximate idea of how the latter probability could change with new data. For fixed d, the samples $\{F_t(d)\}$ also give estimates of the quantiles of such change. This is how the ranges of figs 1, 7, 3 were obtained.

An analogous discussion holds for the marginal and conditional probabilities that we can obtain from $\mathrm{p}(d_0 \mid D)$.

B.4 Assessment of the augmented algorithm's long-run utility yield

Besides making a decision – such as choosing a class – for each new unit, we generally must also decide which algorithm to use for such a future task, among a set of candidates. This latter decision depends on the future performance of each algorithm, which in turn depends on the decision that the algorithm will make for each new unit. Two kinds of unknown accompany this double decision: we do not know the classes

[59] R Core Team 2022. [60] nimble 2021. [61] Dyrland et al. 2022b, scripts `RFmcmc.R` and `NNmcmc.R`. [62] Dyrland et al. 2022b, files `transducer_params-Random_Forest.zip` and `transducer_params-Neural_Net.zip`.

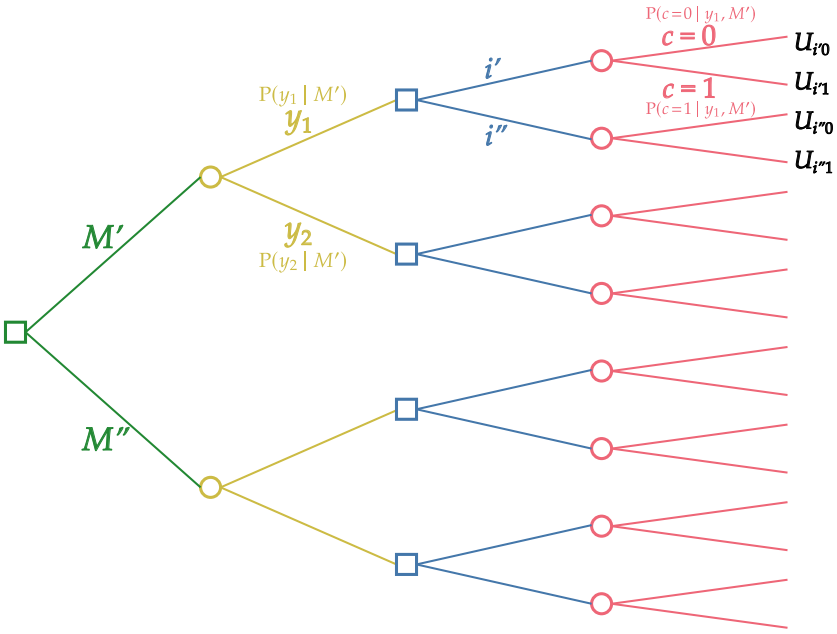

Figure 9 Decision tree for the choice of algorithm. From left to right: Decision node about the algorithm (green), uncertainty node about the algorithm's output (yellow), decision node (e.g. about class) for the inspected item (blue), uncertainty node about the class (red). Only some example nodes and branches are labelled.

of the future units, and we do not know which outputs each algorithm will give for the future data.

This more complex kind of decision & uncertainty problems are also dealt with decision theory. Their theory is presented and applied step-by-step in the humorous lectures by Raiffa 1970 ch. 2; other references are[63]. We here give only a sketch and refer to the works above for details.

Our double decision & uncertainty problem can be represented as a *decision tree*. A very simplified example is illustrated in fig. 9.

We imagine to have to choose between two classification algorithms M' and M''. This choice corresponds to the *decision node* on the left (green). Decision nodes are represented by squares.

If we choose to use algorithm M', then it may happen that it will give either output y_1 or y_2 when applied to a new unit. We are uncertain about

[63] Bernardo & Smith 2000 § 2.2; Pratt et al. 1996; Raiffa & Schlaifer 2000; Luce & Raiffa 1957.

which output will occur, with probabilities $\mathrm{P}(y_1 \mid M')$ and $\mathrm{P}(y_2 \mid M')$. This uncertainty corresponds to an *uncertainty node* (yellow). Uncertainty nodes are represented by circles.

Once the output of the algorithm is known, we must decide (blue decision nodes) among choices i' and i'', for example to choose whether to consider the new unit as class 0 or class 1.

The unit will turn out to be class 0 or class 1: we are uncertain about which (red decision nodes), with probabilities $\mathrm{P}(c=0 \mid y_1, M')$, $\mathrm{P}(c=1 \mid y_1, M')$ if the output was y_1, and with probabilities $\mathrm{P}(c=0 \mid y_2, M')$, $\mathrm{P}(c=1 \mid y_2, M')$ if the output was y_2.

Finally, depending on our choice between i' and i'' and on the actual class, we will gain one of the four utility amounts $U_{i1\,0}$, $U_{i'\,1}$, $U_{i''\,0}$, $U_{i''\,1}$. These are the elements of the utility matrix discussed in § 3; we have seen concrete numerical examples in the demonstration of § 4.

An analogous analysis and probabilities hold if we choose algorithm M'' (the output space of this algorithm can be different from that of M').

The basic procedure of this decision problem is to first calculate expected utilities starting from the terminal uncertainty nodes, making optimal decisions at the immediately preceding decision nodes. Each such decision will therefore have an associated utility equal to its corresponding maximal expected utility. The same procedure is then applied to the uncertainty nodes about the outputs. In this way each algorithm receives a final expected utility; in formulae,

$$\text{utility of algorithm } M = \sum_y \left[\max_i \left\{ \sum_c U_{ic}\, \mathrm{P}(c \mid y, M) \right\} \right] \mathrm{P}(y \mid M)\,. \quad (29)$$

Either sum is replaced by an integral over a density if the related quantity, y or c, is continuous.

What is important in the formula above is that the probabilities for the outputs and the conditional probabilities for the classes given the outputs are known: *they are the ones calculated for the transducer from the calibration set*.

The expected utilities of table 3, § 5.2, were calculated with the formula above (the integral over y being approximated by a sum over a dense grid).

These values are *expected* utilities, though. One may ask: what is the probability that the final utility of one model will actually be higher or lower than the other's?

We can answer this question, again thanks to de Finetti's formula (21), similarly to how we did with the variability of the transducer curves, explained in the previous § B.3. The long-run utility of the algorithm M is given by formula (29) but with the probabilities replaced by the long-term frequencies $F(c \mid y)$ and $F(y)$. The probability of this long-run utility is then determined by the density $w(F)$ represented by a set of samples. Calculating the long-run utility for each sample we can finally construct a probability histogram for each algorithm's utility. The histograms of fig. 6 are obtained this way. From them we can also calculate the probability that an algorithm's utility u' will be higher than another's utility u'', corresponding to the integral

$$\iint \delta(u' > u'') \, \mathrm{p}(u') \, \mathrm{p}(u'') \, \mathrm{d}u' \, \mathrm{d}u'' \,. \tag{30}$$

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.

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

Cheeseman, P. (1988): *An inquiry into computer understanding*. Comput. Intell. **4**[2], 58–66. doi:10.1111/j.1467-8640.1988.tb00091.x.

— (2018): *On Bayesian model selection*. In: Wolpert (2018): 315–330. First publ. 1995.

Chen, H., la Engkvist, Wang, Y., Olivecrona, M., Blaschke, T. (2018): *The rise of deep learning in drug discovery*. Drug Discov. Today **23**[6], 1241–1250. doi:10.1016/j.drudis.2018.01.039.

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.

Correa Bahnsen, A., Aouada, D., Ottersten, B. (2015): *Example-dependent cost-sensitive decision trees*. Expert Syst. Appl. **42**[19], 6609–6619. doi:10.1016/j.eswa.2015.04.042.

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).

de Finetti, B. (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.

de Valpine, P., Paciorek, C., Turek, D., Michaud, N., Anderson-Bergman, C., Obermeyer, F., Wehrhahn Cortes, C., Rodríguez, A., et al. (2021): *Nimble: MCMC, particle filtering, and programmable hierarchical modeling*. https://cran.r-project.org/package=nimble, doi:10.5281/zenodo.1211190, https://r-nimble.org. First publ. 2016.

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.

Elkan, C. (2001): *The foundations of cost-sensitive learning*. In: *Proceedings of the seventeenth international joint conference on artificial intelligence, ijcai 2001*, ed. by B. Nebel (Kaufmann): 973–978. https://www.ijcai.org/Proceedings/01/IJCAI-2001-k.pdf.

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

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).

— (1966): *How to estimate probabilities*. J. Inst. Maths. Applics **2**[4], 364–383.

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).

Goodman, L. A., Kruskal, W. H. (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. (1996): *Sentential Probability Logic: Origins, Development, Current Status, and Technical Applications*. (Associated University Presses, London).

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

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.

Hingorani, A. D., Kuan, V., Finan, C., Kruger, F. A., Gaulton, A., Chopade, S., Sofat, R., MacAllister, R. J., et al. (2019): *Improving the odds of drug development success through human genomics: modelling study*. Sci. Rep. **9**, 18911. doi:10.1038/s41598-019-54849-w.

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.

Ishwaran, H., Zarepour, M. (2002): *Dirichlet prior sieves in finite normal mixtures*. Stat. Sinica **12**[3], 941–963. http://www3.stat.sinica.edu.tw/statistica/J12n3/j12n316/j12n316.htm.

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.

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

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.

Johnson, N. L., Kemp, A. W., Kotz, S. (2005): *Univariate Discrete Distributions*, 3rd ed. (Wiley, New York). First publ. 1969.

Johnson, N. L., Kotz, S., Balakrishnan, N. (1994): *Continuous Univariate Distributions. Vol. 1*, 2nd ed. (Wiley, New York). First publ. 1970.

— (1995): *Continuous Univariate Distributions. Vol. 2*, 2nd ed. (Wiley, New York). First publ. 1970.

— (1996): *Discrete Multivariate Distributions*. (Wiley, New York). First publ. 1969 in chapter form.

Kotz, S., Balakrishnan, N., Johnson, N. L. (2000): *Continuous Multivariate Distributions. Vol. 1: Models and Applications*, 2nd ed. (Wiley, New York). First publ. 1972 by N. L. Johnson and S. Kotz.

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.

Ling, C. X., Sheng, V. S. (2017): *Cost-sensitive learning*. In: Sammut, Webb (2017): 285–289. DOI:10.1007/978-1-4899-7687-1_181.

Luce, R. D., Raiffa, H. (1957): *Games and Decisions: introduction and critical survey*. (Wiley, New York).

Lundervold, A. S., Lundervold, A. (2019): *An overview of deep learning in medical imaging focusing on MRI*. Z. Med. Phys. **29**[2], 102–127. DOI:10.1016/j.zemedi.2018.11.002.

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.

MacKay, D. J. C. (2005): *Information Theory, Inference, and Learning Algorithms*, Version 7.2 (4th pr.) (Cambridge University Press, Cambridge). https://www.inference.org.uk/itila/book.html. First publ. 1995.

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. (1993): *Probabilistic inference using Markov chain Monte Carlo methods*. Tech. rep. CRG-TR-93-1. (University of Toronto, Toronto). http://www.cs.utoronto.ca/~radford/review.abstract.html, https://omega0.xyz/omega8008/neal.pdf.

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.

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.

Pratt, J. W., Raiffa, H., Schlaifer, R. (1996): *Introduction to Statistical Decision Theory*, 2nd pr. (MIT Press, Cambridge, USA). First publ. 1995.

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.

R Core Team (2022): *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org. First released 1995.

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

Raiffa, H., Schlaifer, R. (2000): *Applied Statistical Decision Theory*, repr. (Wiley, New York). First publ. 1961.

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.

Sink, R., Gobec, S., Pečar, S., Zega, A. (2010): *False positives in the early stages of drug discovery*. Curr. Med. Chem. **17**[34], 4231–4255. doi:10.2174/092986710793348545.

Smith, J. E., Winkler, R. L. (2006): *The optimizer's curse: skepticism and postdecision surprise in decision analysis*. Manag. Sci. **52**[3]. doi:10.1287/mnsc.1050.0451.

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.

Wolpert, D. H., ed. (2018): *The Mathematics of Generalization*, repr. (CRC Press, Boca Raton, USA). doi:10.1201/9780429492525. First publ. 1995.