Evaluation and Improvement of Machine Learning Algorithms in Drug Discovery

Kjetil Dyrland

Supervisor: P.G.L. Porta Mana

Co-Supervisor: Alexander S. Lundervold

Master's thesis in Software Engineering at

Department of Computer Science, Electrical Engineering and Mathematical Sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

June 1, 2022

Western Norway University of Applied Sciences

UNIVERSITAS BERGENSIS

Acknowledgements

I would like to express my sincerest gratitude to my supervisor P.G.L. Porta Mana and co-supervisor Alexander S. Lundervold for guiding me through this project and for all the great meetings and discussions, and especially P.G.L. for bringing 'boller' and coffee to the meetings. I would also like to thank my family for endless support throughout the year, and my partner Synne for constant love, support, and encouragement. Thanks to the Mohn Medical Imaging and Visualization Centre where the studies and experiments were performed.

Contents

1 Summary **1**

I Background **3**

2 Overview of Machine Learning **4**

2.1 Introduction to Machine Learning 4
2.2 Strengths and limitations 5
2.3 Dataset and inputs 5
2.4 Overfitting and Underfitting 6
2.5 Evaluation 6
2.6 Algorithm example: Random Forest 8

3 Deep Learning for Computer Vision **9**

3.1 Introduction 9
3.2 Loss functions 10
3.3 Convolutional neural networks for computer vision 12
3.3.1 ResNet 12

4 Drug Discovery **13**

4.1 Introduction 13
4.2 Representations in Drug Discovery 14

5 Overview of Part II **17**

5.1 Does the evaluation stand up to evaluation? 17
5.1.1 Context 17
5.1.2 Results 17
5.2 A probability transducer for machine-learning classifiers 18
5.2.1 Context 18
5.2.2 Results 18

Bibliography **19**

II Publications **21**

6 Does the evaluation stand up to evaluation? **22**

7 A probability transducer and decision-theoretic augmentation for machine-learning classifiers **58**

List of Figures

2.1 Confusion matrix for binary classification 7

3.1 Max pooling with 2x2 stride 10
3.2 Structure of an artificial neural network 11

4.1 Generation of 2D depictions 15

List of Papers

6 Does the evaluation stand up to evaluation? 22
7 A probability transducer and decision-theoretic augmentation for machine-learning classifiers 58

Chapter 1

Summary

Drug discovery plays a critical role in today's society for treating and preventing sickness and possibly deadly viruses. In early drug discovery development, the main challenge is to find candidate molecules to be used as drugs to treat a disease. This also means assessing key properties that are wanted in the interaction between molecules and proteins. It is a very difficult problem because the molecular space is so big and complex. Drug discovery development is estimated to take around 12–15 years on average, and the costs of developing a single drug amount to $2.8 billion dollars in the US [1].

Modern drug discovery and drug development often start with finding candidate drug molecules ('compounds') that can bind to a target, usually a protein in our body. Since there are billions of possible molecules to test, this becomes an endless search for compounds that show promising bioactivity. The search method is called high-throughput screening (HTS), or virtual HTS (VHTS) in a virtual environment. The traditional approach to HTS has been to test every compound one by one. More recent approaches have seen the use of robotics and of features extracted from the molecule, combining them with machine learning algorithms, in an effort to make the process more automated. Research has shown that this will still lead to human errors and bias [2, 3]. So, how can we use machine learning algorithms to make this approach more cost-efficient and more robust to human errors?

This project tried to address these issues and led to two scientific papers as a result. The first paper explores how common evaluation metrics used for classification can actually be unsuited to the task, leading to severe consequences when put into a real application. The argument is based on basic principles of Decision Theory, which is recognized in the field of machine learning [4] but has not been put into much use. It makes a distinction between predicting the most probable class and predicting the most valuable class in terms of the "cost" or "gains" for the classes. In an algorithm for classifying a particular disease in a patient, the wrong classification could lead to a life or death situation. The principles also apply to drug discovery, where the cost of further developing and optimizing a "useless" drug could be huge [5, 6]. The goal of the classifier should therefore not be to guess the correct class but to choose the *optimal* class, and the metric must depend on the type of classification problem. Thus,

we show that common metrics such as precision, balanced accuracy, F1-score, Area Under The Curve, Matthews Correlation Coefficient, and Fowlkes-Mallows index are affected by this problem, and propose an evaluation method grounded on the foundations of Decision Theory to provide a solution to this problem. The metric presented, called *utility*, takes into account gains and losses for each correct or incorrect classification of the confusion matrix. For this to work effectively, the output of the machine learning algorithm needs to be a set of sensible probabilities for each class. This brings us to the second paper.

Machine learning algorithms usually output a set of real numbers for the classes they try to predict, which, possibly after some transformation (for example the 'softmax' function), are meant to represent probabilities for the classes. However, the problem is that these numbers cannot be reliably interpreted as actual probabilities, in the sense of degrees of belief [7]. In the paper, we propose the implementation of a probability transducer to transform the output of the algorithm into sensible probabilities. These are then used in conjunction with the utilities to choose the class with the *maximal expected utility*. The results show that the transducer gives better scores, in terms of the utilities, for all cases compared to the standard method used in machine learning.

Part I

Background

Chapter 2

Overview of Machine Learning

This section will give a quick overview of some basic machine learning (ML) concepts and techniques. They must be understood before we go into more detail in the rest of the thesis.

2.1 Introduction to Machine Learning

Machine learning is a sub-field of artificial intelligence (AI) and is based on statistics, probability, mathematics, and computer science. Its purpose is to learn deterministic or statistical relationships from data, as well as the features in the data that are important for such relationships, and to make decisions based on the learned knowledge. A machine learning model is an algorithm that is trained on a specific set of data. Machine learning has three main categories based on the purpose of the algorithm:

- In **supervised learning**, the purpose is to find the unknown value of something, for example, an object's class or a physical quantity, given the known values of other variables. For this reason, the training data is labeled, meaning the data has a "ground truth" value and the model tries to predict this value. We call the prediction classification if the goal is to predict what class the input belongs to, e.g. a cat or a dog. We call the prediction regression if the value to be predicted can assume a continuous range of values, e.g. the price of a house.

- **Unsupervised learning** tries to find features and relations in the data, which are therefore unlabelled during training. Since the data are unlabelled there is no right or wrong answer that a human can make much sense of. This also makes it difficult to evaluate the performance of machine learning algorithms in this category.

- **Reinforcement learning** uses the concept of giving rewards and penalties as feedback signals to an algorithm. Based on the feedback signals, the algorithm learns the best method for the problem. Reinforcement learning

is used in tasks like self-driving cars, automatic drones, and video game AIs.

2.2 Strengths and limitations

One of the benefits of using ML is that the computational part is automated. It takes away the stress from the human that can lead to human errors and takes care of repetitive and computationally expensive procedures. ML can also find trends and patterns that are not apparent to humans. For example, data that appear in high-dimensional spaces can be difficult for humans to interpret. It can also be used to solve problems in a wide area of fields, from self-driving cars to the prediction of house prices. Advancements in computing power and the availability of good software in recent years have made ML an extremely cheap and fast option for research tasks [8].

All this said, ML still has a lot of limitations. First of all, training an ML model requires a lot of data, and in some fields data are scarce. Second, the data have to be representative of the future data that the ML algorithm will process during its operation, and not be affected by biases or other errors. Third, there is also an ethical problem: when the ML algorithm makes a mistake, who is to blame for that mistake? Should the programmer be blamed, the computer, or someone else? For example, in medicine and drug discovery, an erroneous prediction by the ML model could lead to giving wrong diagnoses or wrong drugs to patients and could, in the worst case, be life-threatening.

Fourth, there is the problem of tuning and evaluating the model. Tuning the model is no easy process, as there is a bunch of small parameters to tweak and adjust. Getting these wrong often leads to unwanted bias or overfitting problems. There is also an abundance of evaluation metrics to choose from; we will explore some in section 2.5. Usually, all of them are used for the same numerical experiments even though they were made for completely different use cases. Many of the commonly used ML evaluation metrics in classification are not reliable when looking for the most optimal algorithm [9, 10].

2.3 Dataset and inputs

ML algorithms depend on the quality and quantity of the dataset they are given for learning. If the dataset is poorly put together and small, the algorithm will most likely perform poorly. That is why ML practitioners say that working with machine learning algorithms is at least 80% data pre-processing and cleaning and 20% algorithms [11]. Some of the more difficult problems in data pre-processing include identifying erroneous values, compensating for missing values, and estimating noise. Most of the time, it is not possible to solve these problems directly, so the programmer has to work around them. For data points with partially missing values, for example, the programmer may simply discard the whole data point, or use estimated values in place of the missing ones. Data augmentation is a set of techniques to create synthetic but plausible data, to increase the amount of training data. For example, in image-related problems, data augmentation consists of creating additional images by rotating, flipping, or modifying colors in an image of the original dataset. For instance, from an

image of a dog, we can create images with the same dog but turn it upside-down or use slightly different colors. The idea behind such a procedure is to help the ML algorithm learn which image features are relevant (such as specific shapes), and which are irrelevant (such as positions or orientations).

2.4 Overfitting and Underfitting

In all ML applications, the model must be effective on new, unseen data, meaning it has to generalize well. If the performance is poor on new unseen data, it usually means that the model has either overfitted or underfitted on the training data. Overfitting is the case where the model relies on features in the training data that are irrelevant to any unseen data. Underfitting is the tendency to not learn specific features that are relevant and present in the training data.

So, how do we solve these problems?

First of all, it is common practice to split the dataset into a training, validation, and test set. The training data is used to fit the model; the validation set is used to validate the model and update its hyperparameters (learning rate, regularization, etc.). The test set is a 'final' test to check how good the model is for 'unseen' data. Doing this allows us to see more easily if the model is overfitting or underfitting, by looking at the loss and accuracy for the training and validation sets. The model overfits if the training loss goes significantly down and validation loss goes up after each epoch. Conversely, the model underfits if the loss stays high for both the training data and validation data. To solve the overfitting problem, the best solution would be to collect more training data, but in most cases, this is not an option. Other options can be to apply techniques, such as regularization, to the model (L1, L2, dropout, etc.). Regularization is a technique that can be applied to the model to punish it for being too flexible.

For underfitting, the best approach is to use a more flexible model. If this approach can't be applied or doesn't work, techniques such as feature engineering could work. Feature engineering is the process of extracting features from the raw data that the model can more easily interpret.

All in all, the general method to prevent generalization error is to train a model that is not too complex or too simple.

2.5 Evaluation

Evaluation is an essential part of any machine learning project. After the model is trained, we need to test its performance. As explained in the previous section, evaluation is done on an independent test set. The ultimate goal is usually to apply the ML model to production. For this reason, it is important to choose the right metric to evaluate the performance of the model. And what is "right" depends on the task.

For simplicity, let's consider the evaluation of a binary-classification algorithm, that is, an algorithm that has to decide whether to classify each new data point into one of two possible classes. Conventionally one class is called 'positive' and the other 'negative'. The whole performance of the model on the test set is embodied in four statistics:

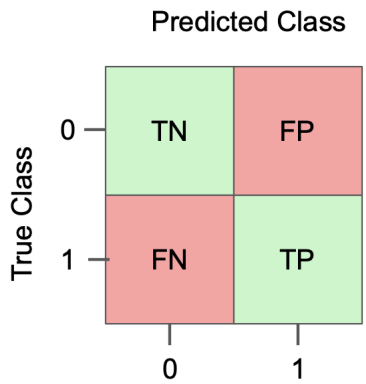


Figure 2.1: Confusion matrix for binary classification

- True positives (TP): the number of cases in which the model chose the positive class, and the true class was indeed the positive one.
- False positives (FP): the number of cases in which the model chose the positive class, but the true class was the negative one.
- True negative (TN): the number of cases in which the model chose the negative class, and the true class was indeed the negative one.
- False negatives (FN): the number of cases in which the model chose the negative class, but the true class was the positive one.

These four statistics are collected in a 2-by-2 table or matrix, commonly called a 'confusion matrix'. The true classes constitute its rows, and the predicted classes are its columns, but can also be displayed with the predicted classes as its rows and the true classes as its columns (as in our papers).

The best theoretical performance where all instances are correctly classified based on their 'ground truth' label will have a confusion matrix with all nonzero elements on the diagonal from the top left corner to the bottom right. Figure 2.1 shows the representation of a confusion matrix for binary classification where TN and TP lie on the diagonal.

Some traditional metrics for binary classification are:

- *Accuracy*:
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
- *Precision*:
$$Precision = \frac{TP}{TP\ +\ FP}$$
- *Recall*:
$$Recall = \frac{TP}{TP\ +\ FN}$$
- *F1 score*:
$$F1 = 2\ \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$$

- *Matthews correlation coefficient (MCC):*

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Precision shows the ability of the model to not label a negative example as positive. *Recall* shows the ability of the model to label positive samples. Recall and precision have a linear relationship, meaning most of the time they are not both high at the same time. An increase in precision will decrease recall and vice versa. Because of this, it is important to look at the task to choose one or the other.

The *F1 score* finds a balance between these two metrics, by taking their harmonic mean.

Matthews correlation coefficient (MCC) compares the correlation between the actual class and the predicted class. It scales from -1 to +1 where 0 indicates a random correlation between the two. MCC is considered to be a balanced evaluation metric and became known in the ML community from the work by Matthews in [12].

These are all traditional evaluation metrics commonly used in binary classification. However, they can be misleading and can lead to overoptimistic results as we will show and discuss in the included paper. For a brief overview and more examples, consult also the paper by P. Flach [9].

For a regression task, we need something that can measure how close the prediction are to the real number. Some of the most common metrics in regression task are *mean squared error* (MSE):

$$L(x, y) = \sum_{i=1}^{D} (x_i - y_i)^2$$

And the root mean squared error (RMSE):

$$L(x, y) = \sqrt{MSE}$$

Where x is the actual value and y is the predicted value.

2.6 Algorithm example: Random Forest

Random forest (RF) is a machine learning algorithm based on the ensembling of decision trees [13]. RF has gained a lot of popularity in recent years due to its ability to handle a large number of features, its good performance, and its resilience to overfitting. Such models have already delivered powerful results in compound classification and QSAR analysis [14]. The last layer in an RF classifier, the output layer, bases the decision on votes from each tree for each class, then rescales the output to be between 0 and 1.

Chapter 3

Deep Learning for Computer Vision

3.1 Introduction

Deep learning (DL) is a sub-field of machine learning and is part of a class of artificial neural networks (ANN). DL algorithms have complex architectures which consist of several layers of processing elements (input, hidden and output layers, etc.) [15]. The idea of DL is decades old but has become widely popular in recent years thanks to the availability of computational power and data that it demands. There are several popular DL models, the most common being:

- *MLP:* Multilayer perceptron (MLP) is the base architecture of deep learning or feed-forward artificial neural networks. It consists of a network with an input layer, one or more hidden layers, and an output layer, with full connections between successive layers, as seen in Figure 3.2. Each node in a network is connected to the next layer with an assigned weight. The weights are randomized before training but get updated to learn from the data and correctly predict the true value. The weights get updated using backpropagation and gradient descent [8].

- *CNN or ConvNet:* A convolutional neural network (CNN) is a network that consists of convolutions, pooling layers as well as fully connected layers.

  The building blocks of a CNN are:

  - *Convolutional layer:* A convolution takes advantage of the two-dimensional (2D) structure of the data and computes a feature map. This is done by multiplying a set of weights, called a kernel, over the 2D space of the input which then produces an output with a single value. CNN is great at detecting important features in 2D space and is, and the standard deep learning network to use when it comes to image and video tasks.
  - *Activation layer:* The activation layer is used to provide a non-linearity in the network. This function is usually a rectified linear

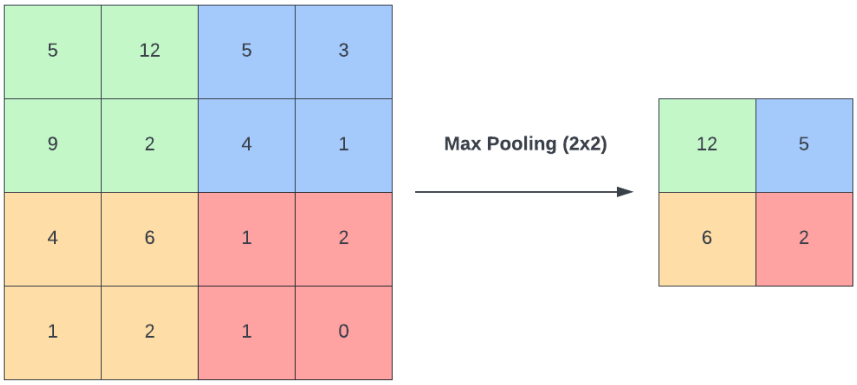


Figure 3.1: Max pooling with 2x2 stride

unit or ReLU. The ReLU function has become the most popular and widely used in the last years due to its ability to significantly accelerate training compared to other activation functions. The ReLU function makes all negative numbers zero and is defined as $F(x) = max(0, x)$.

- *Pooling layer:* The pooling layer is often used in CNNs to reduce the feature map created by the convolutional layer, saving computational power and memory requirements. The two most common pooling techniques are max pooling and average pooling. Max pooling works by taking the max of each grid in the feature map, as illustrated in figure 3.1. Average pooling works by taking the average inside each grid. The pooling layer does not have any learnable parameters, which could mean valuable features are lost in the process.
- *Output layer:* The output layer is the last layer in a CNN. In classification, it outputs a set of values for each class, then transformed to values that should better represent probabilities for each class using a *sigmoid*, *softmax* or *tanh* functions. The sigmoid function is a logistic regression function for binary classification and outputs a set of numbers between 0 and 1. softmax works similarly to sigmoid but works for multiclass classification as well, and the output sums to 1. *Tanh* works the same as sigmoid but outputs numbers between -1 and 1.

3.2 Loss functions

The loss function in a deep neural network (DNN) measures how well the model predicted the correct outputs for the given input. The loss function is used to decide how to update the weights in a stochastic gradient descent (SGD) optimization step, the learning process in DNNs, to better fit the data. The loss function is chosen based on the task. Here are some common loss functions for classification and regression tasks.

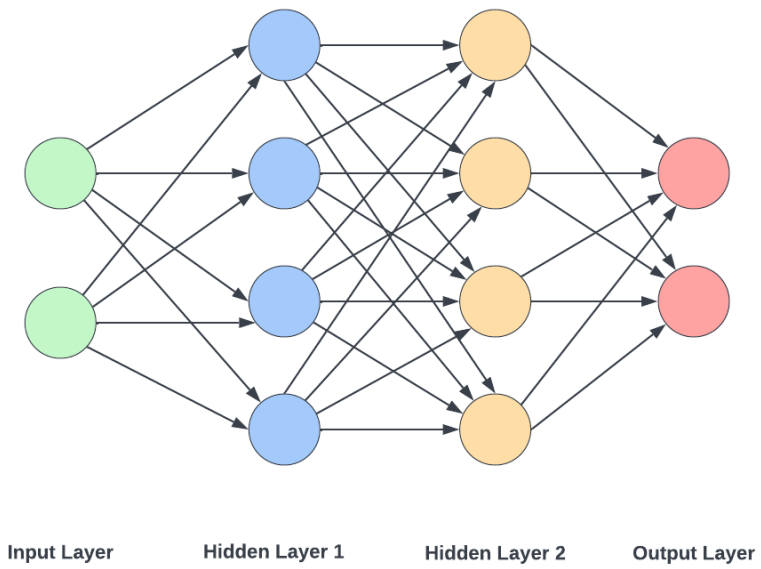


Figure 3.2: Structure of an artificial neural network

Cross-Entropy Loss

Cross-entropy loss is often used in classification tasks. Cross-entropy loss can be defined as:

$$CE = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

The output of this loss function is a number between 0 and 1, where higher values mean a higher loss/error in the prediction.

Focal loss

Focal loss is a dynamically scaled cross-entropy loss that can handle class imbalance in the dataset. It tries to compensate for imbalanced datasets by increasing the loss for classes with the lowest amount of data-points available. Focal loss is defined as:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

By choosing a value $\gamma > 0$ it reduces the loss for easily classified examples where $p_t > 0.5$ and focuses more on misclassified examples.

RMSE

Root mean squared error (RMSE) is a loss function used for regression tasks. It is the root-mean-square average of the differences between predicted and actual

values. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

3.3 Convolutional neural networks for computer vision

Computer vision is a sub-field of artificial intelligence, described as the ability of a computer to extract meaningful features from videos, images, or other visual inputs. The breakthrough in deep learning for computer vision has become one of the most important fields in the last years, thanks to the use of CNNs and techniques such as pre-training.

3.3.1 ResNet

Residual neural network (ResNet) is a particularly deep and complex form of CNN architecture. It was proposed by He et al. [16] and won the ImageNet detection and visualization challenge in 2015 [17]. It adds so-called skip connections to the more standard architectures of its time. A skip connection allows the network to skip unnecessary layers. In practice this means that a network with 50 layers should perform at least as well as one with 20 layers, skipping the last 30 layers if necessary. ResNet is to this day seen as one of the best performing CNNs for solving many computer vision problems.

Chapter 4

Drug Discovery

4.1 Introduction

Simplifying things a little, a drug is a molecule, or compound, that fits into a "hole" in a protein, called a target, in our body. The combination of the two leads to a subtle structure change in the protein, inducing other chemical reactions, possibly in a domino chain of reactions [11]. These reactions are expected to have positive consequences for the organism's health. When a compound has the capability of combining with a target, it is said to be 'active', or we say there 'there is activity' between the two.

The process of drug discovery has five critical stages:

- **Target selection and validation**:
  The process of identifying relevant proteins that exist in the body and that can be used as targets for compounds.
- **Hit discovery:**
  Hit discovery focuses on screening millions of compounds to find activity when binding to a target. High throughput screening (HTS) is the search for finding this type of molecule, which is then called a 'hit'. Virtual HTS (VHTS) uses software to accomplish the same thing.
- **Hit to lead**:
  This process focuses on optimizing the hit to increase the strength of the binding between the drug molecule and the target (affinity).
- **Lead optimization**:
  Further optimization to discover viable drug candidates by looking for wanted properties, known as ADMET properties: absorption, distribution, metabolism, excretion, and toxicity. Toxicity, in this case, means the drug binds to other targets than the intended one.
- **Pre-clinical development**:
  The stage before clinical trials includes testing in a wet lab and on animals.

Clinical trials are the last stage and are a precondition for the drug to be approved by the government.

4.2 Representations in Drug Discovery

There are a plethora of options to represent a molecule. In physical space, a molecule is just a group of atoms kept together by particular physical forces or 'bonds', with unique chemical features. A molecule is incredibly small and always in motion; so, how can we represent a molecule with all the unique chemical features and structures still visible to the computer?

Graphs

For a small molecule, graphs can be used. Atoms are represented as nodes and the bonds between them as edges in the graph. It is a 2D representation that is utilized to represent 3D information. Nodes in a graph do not have spatial relationships, only topological relationships. The topological relationship is encoded as a node and/or edge attribute in the graph. The resulting graph also depends on the algorithm used to traverse the spatial structure of the molecule. The order of atoms and alignment will be different depending on the algorithm. Furthermore, when working with molecular graphs, there is no one correct way to represent any molecule, and the representation chosen must be appropriate for the task [18].

SMILES

The Simplified Molecular Input Line Entry System (SMILES) was developed in 1988 by Weininger et al. [19] and has been a very popular molecular representation since then. It is a single string of letters and is based on a graph representation of the molecule. The SMILES are non-unique, representing each atom that is traversed in the graph in order. In RDKit [20] the algorithm uses a depth-first search.

Fingerprints

Another representation, chemical descriptors, can be used to describe certain physicochemical, structural, topological, and/or electronic properties of a compound. A common way to do this is hashed circular fingerprints, which are representations of chemical structures by atom neighborhoods and have been widely applied in Quantitative Structure-Activity Relationship (QSAR) analysis. A widely used class of circular fingerprints is ECFP (Extended Connectivity Fingerprints) [21], based on the Morgan algorithm.

2D Depictions

Depictions can be used for direct visualization of compounds and their physicochemical properties. 2D depictions are a vector image representation of the skeletal structure of the molecule. As with any representation, 2D depictions also come with difficulties (orientation, overlap, rendering, etc.). Some difficulties can be overcome with the use of algorithms. In this project, we use the RDKit algorithm, which addresses a lot of those problems. The generation of 2D depictions from SMILES is illustrated in figure 4.1 Molecules can also be depicted in various ways for describing interactions and reactions.

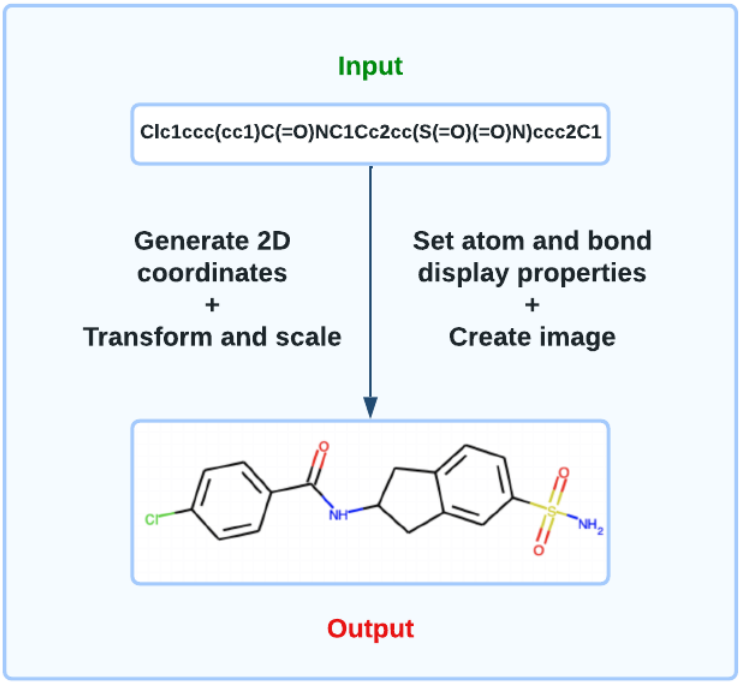


Figure 4.1: Generation of 2D depictions

3D Depictions

Software can also be used to represent and visualize molecules and interactions in a 3D space. Popular representations are *ball-and-stick*, *cartoon*, and *van der Waals* (vdW). Each comes with its own useful visualization of specific properties. For example, vdW spheres can be used to visualize interactions between protein and ligands. Because these interactions are more complex, 3D depictions are especially useful in docking and mechanistic studies, whereas 2D depictions are more useful in a less complex molecule space as in structure-activity predictions studies [18].

Chapter 5

Overview of Part II

The core of this thesis consists of two papers that are connected to the study of machine learning in drug discovery. The purpose of this chapter is to present the papers with the current knowledge of the background information presented in Part 2. In other words, additional information, context, and results are discussed.

5.1 Does the evaluation stand up to evaluation?

5.1.1 Context

From the background information in chapter 4, it is clear that the drug discovery process is extremely expensive and time-consuming, and that machine learning algorithms have the potential and ability to alleviate this problem. For that to be realized, the evaluation of machine learning models has to be improved [9].

In the paper, it is argued that evaluation metrics used in machine learning can be misleading when put into a real-life application. A machine learning algorithm that is favored by a majority of evaluation metrics can still turn out to not be the best one in practice. It is proposed that the evaluation of machine learning models must be grounded based on the principles of Decision Theory [4, 22].

5.1.2 Results

In the paper, we show that the metric must depend on the type of classification problem. A metric based on so-called *utilities* takes into account gains and losses for each correct or incorrect classification. To calculate this we need to specify the cost, or *utility*, for each class of (choice, true class) in the confusion matrix. The *utility yield* is the sum of the product of utilities and the confusion matrix of a classifier. The utility yield is a linear combination of the elements of the confusion matrix, so for any metric to agree with the utility yield it has to be a linear combination of the confusion matrix elements. We show that common metrics such as precision, F1-measure, Matthews correlation coefficient, balanced accuracy, and Fowlkes-Mallows index are not linear combinations of the confusion matrix. This means that these metrics are affected by cognitive

biases and that there is no classification problem where they can correctly rank the performance of all pairs of algorithms.

We also show that using incorrect utilities, with errors as high as 20% of the maximal utility, will still lead to more correctly ranked pairs than the use of any other metric.

5.2 A probability transducer for machine-learning classifiers

5.2.1 Context

This paper explains the important difference between a typical 'cat vs dog' image classification on one side, and, on the other side, classifying a particular disease in a medical application, or classifying a compound as active or not in a drug discovery application. In drug classification, specifically VHTS, we know that the cost and time constraints of wrongly classifying a drug as a false positive can be huge [5, 6]. The goal of a classifier should not be to guess the correct class, but rather to choose the *optimal* class.

5.2.2 Results

The paper introduces an implementation of a 'probability transducer' for generating *sensible probabilities* from the algorithm output. The probabilities are then used together with the utilities to choose the class with maximal expected utility. This method, which we call 'augmentation', will ensure that the classifier picks the most optimal class. The transducer has a low computational cost and allows us to not make any structural changes in the algorithm architecture or the training process. It also gives us an evaluation score of the future overall performance of the classifier, to compare it to other classifiers. To illustrate our method we trained two different classifiers, a random forest, and a convolutional neural network, on the ChEMBL [23] dataset. The data consists of structure-activity relationships, where the classifiers predict 0 for 'inactive' compounds and 1 for 'active' compounds. The models use different pre-processing methods for the data, ECFP for the random forest, and 2D depictions for the convolutional neural network. Both models, according to commonly used metrics, perform at the level of recent state-of-the-art models in VHTS [24]. Considering a large number of possible utility matrices, we show that the utility yield for both models increases in basically all cases. The increase is most notable in the cases where the performance is particularly high or low. The most important utilities for drug discovery is when the FP has a relatively low score compared to TP, as discussed earlier. For utilities where FP is low and TP high, the augmentation gives a near-optimal utility yield and a huge relative increase over the standard method.

Bibliography

[1] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[2] Jochen Sieg, Florian Flachsenberg, and Matthias Rarey. In need of bias control: evaluating chemical data for machine learning in structure-based virtual screening. *Journal of chemical information and modeling*, 59(3): 947–961, 2019.

[3] Christian Merkwirth and Thomas Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *Journal of chemical information and modeling*, 45(5):1159–1168, 2005.

[4] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson series in artificial intelligence. Pearson, Harlow, UK, fourth global ed. edition, 2022.

[5] Roman Sink, Stanislav Gobec, S Pecar, and Anamarija Zega. False positives in the early stages of drug discovery. *Current medicinal chemistry*, 17(34): 4231–4255, 2010.

[6] Aroon D Hingorani, Valerie Kuan, Chris Finan, Felix A Kruger, Anna Gaulton, Sandesh Chopade, Reecha Sofat, Raymond J MacAllister, John P Overington, Harry Hemingway, et al. Improving the odds of drug development success through human genomics: modelling study. *Scientific reports*, 9(1):1–25, 2019.

[7] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992.

[8] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21, 2021.

[9] Peter Flach. Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9808–9814, 2019.

[10] Kjetil Dyrland, Alexander Selvikvåg Lundervold, and Piero Giovanni Luca Porta Mana. Does the evaluation stand up to evaluation?: A first-principle approach to the evaluation of classifiers, 2022.

[11] S Schroedl. Machine learning for drug discovery in a nutshell—part I, 17 june 2018, 2019.

[12] Brian W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[15] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[18] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in AI-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.

[19] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[20] Greg Landrum. Rdkit documentation. *Release*, 1(1-79):4, 2013.

[21] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[22] D Warner North. A tutorial introduction to decision theory. *IEEE transactions on systems science and cybernetics*, 4(3):200–210, 1968.

[23] A Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J Bellis, Jon Chambers, Mark Davies, Felix A Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, et al. The ChEMBL bioactivity database: an update. *Nucleic acids research*, 42(D1):D1083–D1090, 2014.

[24] Alexios Koutsoukas, Keith J Monaghan, Xiaoli Li, and Jun Huan. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of cheminformatics*, 9(1):1–13, 2017.

Part II

Publications

Chapter 6

Does the evaluation stand up to evaluation?

Open Science Framework DOI:10.31219/osf.io/7rz8t

Does the evaluation stand up to evaluation?

A first-principle approach to the evaluation of 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

27 May 2022; updated 1 June 2022

How can one meaningfully make a measurement, if the meter does not conform to any standard and its scale expands or shrinks depending on what is measured? In the present work it is argued that current evaluation practices for machine-learning classifiers are affected by this kind of problem, leading to negative consequences that appear when classifiers are put to real use and that could have been avoided. It is proposed that evaluation be grounded on Decision Theory, and the consequences of such foundation are explored. The main result is that every evaluation metric must be a linear combination of confusion-matrix elements, with coefficients – 'utilities' – that depend on the specific classification problem. For binary classification, the space of such possible metrics is effectively two-dimensional. It is shown that popular metrics such as precision, balanced accuracy, Matthews Correlation Coefficient, Fowlkes-Mallows index, F_1-measure, and Area Under the Curve are never optimal: they always give rise to an avoidable fraction of incorrect evaluations. This fraction is larger than would be caused by the use of a decision-theoretic metric with moderately wrong coefficients.

0 Prologue: a short story

The manager of a factory which produces a sort of electronic component wishes to employ a machine-learning classifier to assess the durability of each produced component. The durability determines whether the component will be used in one of two possible kinds of device. The

classifier should take some complex features of the component as input, and output one of the two labels '0' for 'long durability', or '1' for 'short durability', depending on the component type.

Two candidate classifiers, let us call them A and B, are trained on available training data. When employed on a separate evaluation set, they yield the following confusion matrices, written in the format

$$\begin{array}{cc} & \textit{true class} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} \textit{classifier output} \\ \begin{array}{c} 0 \\ 1 \end{array} \end{array} & \begin{bmatrix} \text{True } 0 & \text{False } 0 \\ \text{False } 1 & \text{True } 1 \end{bmatrix} \end{array}$$

and normalized over the total number of evaluation data:

$$\text{classifier A:} \quad \begin{bmatrix} 0.27 & 0.15 \\ 0.23 & 0.35 \end{bmatrix}, \tag{1}$$

$$\text{classifier B:} \quad \begin{bmatrix} 0.43 & 0.18 \\ 0.07 & 0.32 \end{bmatrix}. \tag{2}$$

These matrices show that the factory produces, on average, 50% short- and 50% long-durability components.

The confusion matrices above lead to the following values of common evaluation metrics[1] for the two classifiers. Class 0 is 'positive', 1 'negative'. **Blue bold** indicates the classifier favoured by the metric, red the disfavoured:

Table 1

Metric	classifier A	classifier B
Accuracy (also balanced accuracy)	0.62	**0.75**
Precision	0.64	**0.70**
F_1 measure	0.59	**0.77**
Matthews Correlation Coefficient	0.24	**0.51**
Fowlkes-Mallows index	0.59	**0.78**
True-positive rate (recall)	0.54	**0.86**
True-negative rate (specificity)	**0.70**	0.64

The majority of these metrics favour classifier B, some of them by quite a wide relative difference. Only the true-negative rate favours classifier A, but only by a relative difference of 9%.

[1] Balanced accuracy: Brodersen et al. 2010; F_1 measure: van Rijsbergen 1974; Matthews correlation coefficient: Matthews 1975; Fowlkes-Mallows index: Fowlkes & Mallows 1983.

The developers of the classifiers therefore recommend the employment of classifier B.

The factory manager does not fully trust these metrics, asking, "how do I know they are appropriate?". The developers assure that these metrics are widely used. The manager (of engineering background) comments, "I don't remember 'widely used' being a criterion of scientific correctness – not after Galileo at least", and decides to employ both classifiers for a trial period, to see which factually leads to the best revenue. The two classifiers are integrated into two separate but otherwise identical parallel production lines.

During the trial period, the classifiers perform according to the classification statistics of the confusion matrices (1) and (2) above. At the end of this period the factory manager finds that the average net gains per assessed component yielded by the two classifiers are

$$\begin{aligned} \text{classifier A:} \quad & \mathbf{3.5}\,€/\text{component}\,, \\ \text{classifier B:} \quad & -3.5\,€/\text{component}\,. \end{aligned} \tag{3}$$

That is, classifier B actually led to a *loss* of revenue. The manager therefore decides to employ classifier A, commenting with a smug smile that it is always unwise to trust the recommendations of developers, unacquainted with the nitty-gritty reality of a business.

The average gains above are easy to calculate from some additional information. The final net gains caused by the correct or incorrect classification of one electronic component are as follows:

$$\begin{array}{cc} & \textit{true class} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \textit{classifier output} \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 15\,€ & -335\,€ \\ -35\,€ & 165\,€ \end{bmatrix} \end{array} \tag{4}$$

The reason behind these values is that short-durability components (class 1) provide more power and are used in high-end, costly devices; but they cause extreme damage and consequent repair costs and refunds if used in devices that require long-durability components (class 0). Long-durability components provide less power and are used in low-end, cheaper devices; they cause some damage if used in devices that require short-durability components, but with lower consequent costs.

Taking the sum of the products of the gains above by the respective percentages of occurrence – that is, the elements of the confusion matrix

– yields the final average gain. The final average gain returned by the use of classifier A, for example, is

$$15\,€ \times 0.27 - 335\,€ \times 0.15 - 35\,€ \times 0.23 + 165\,€ \times 0.35 = 3.5\,€\,.$$

In the present case, the confusion matrices (1) and (2) lead to the amounts (3) found by the manager.

1 Issues in the evaluation of classifiers

The story above illustrates several well-known issues of currently popular evaluation procedures for machine-learning classifiers:

(a) We are swept by an avalanche of possible evaluation metrics. Often it is not clear which is the most compelling. In the story above, for example, one could argue that the true-negative rate was the appropriate metric, in view of the great difference in gains between correct and wrong classification for class 1, compared with that for class 0.

But at which point does this qualitative reasoning fail? Imagine that the net gains had been as follows instead:

$$\begin{array}{cc} & \textit{true class} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} \textit{classifier output} \\ \begin{array}{c} 0 \\ 1 \end{array} \end{array} & \begin{bmatrix} 45\,€ & -335\,€ \\ -65\,€ & 165\,€ \end{bmatrix} \end{array}\,. \tag{5}$$

One could argue that also this case there is a great economic difference between correct and wrong classification for class 1, as compared with class 0. The true-negative rate should, therefore, still be the appropriate metric. Yet a simple calculation shows that, in this case, it is classifier B that actually leads to the best average revenue: 7.3 €/component, vs 4.7 €/component for classifier A. Hence the true-negative rate is *not* the appropriate metric here and our qualitative reasoning failed us.

(b) A classifier favoured by the majority of available metrics can still turn out *not* to be the best one in practice.

(c) Most popular metrics are introduced by intuitive reasoning, ad hoc mathematical operations, special assumptions (such as gaussianity[2]), and analysis of special cases. Unfortunately such derivations

[2] e.g. Fisher 1963 § 31 p. 183 for the Matthews correlation coefficient.

do not guarantee generalization to all cases, nor that the proposed metric is uniquely determined by the chosen assumptions, nor that it satisfies other basic but neglected requirements. By contrast, compare for instance the derivation of the Shannon entropy[3] as the *unique* metric universally satisfying a set of general, basic requirements for the amount of information; or the derivation of the probability calculus[4] as the *unique* set of rules satisfying general rational requirements for inductive reasoning, learning, and prediction[5].

(d) Let us assume that some of the popular metrics identify the best algorithm 'in the majority of cases' – although it is difficult to statistically define such a majority, and no real surveys have ever been conducted to back up such an assumption. Yet, do we expect the end-user to simply *hope* not to belong to the unlucky minority? Is such uncertainty inevitable?

We cannot have a cavalier attitude towards this problem: life and death can depend on it in some machine-learning applications[6]. Imagine a story analogous to the factory one, but in a medical setting instead. The classifiers should distinguish between two tumour types, requiring two different types of medical intervention. The confusion matrices are the same (1) and (2). In the present case, correct or incorrect classification lead to the following expected remaining life lengths[7] for patients in a specific age range:

$$\begin{array}{cc} & \textit{true class} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} \textit{classifier} \\ \textit{output} \end{array} \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 350\,\text{months} & 0\,\text{months} \\ 300\,\text{months} & 500\,\text{months} \end{bmatrix} \end{array} . \tag{6}$$

These values might arise in several scenarios. For example, tumours of class 0 and 1 may require very different kinds of treatment. If a class 0 tumour is misdiagnosed and not properly treated, it leads to immediate death (0 months); if correctly diagnosed, its

[3] Shannon 1948; Woodward 1964 § 3.2; also Good & Toulmin 1968. [4] Cox 1946; Fine 1973; Jaynes 2003 chs 1–2. Some literature cites Halpern 1999a as a critique of Cox's proof, but curiously does not cite Halpern's 1999b partial rebuttal of his own critique, as well as the rebuttals by Snow 1998; 2001. [5] Self & Cheeseman 1987; Cheeseman 1988; Russell & Norvig 2022 ch. 12. [6] cf. Howard 1980. [7] cf. the discussion in Sox et al. 2013 § 11.2.9.

treatment is usually successful, leading to high life expectancy (500 months). Class 0 tumours can be treated, but they lead to a shorter life expectancy (350 months). If they are misdiagnosed as class 1, however, the damage caused by class 1 treatment shortens this life expectancy even further (300 months).

This matrix above is numerically equivalent to (4) up to a common additive constant of 335, so the final net gains are also simply shifted by this amount. It is easy to see that the metrics are exactly as in Table 1, the majority favouring classifier B. And yet the use of classifier A leads to a more than six-month longer expected remaining life than classifier B.

(e) Often it is not possible to temporarily deploy all candidate classifiers, as our fictitious manager did, in order to observe which factually leads to the best results. Or it may even be unethical: consider a situation like the medical one above, where a classifier may lead to a larger number of immediate deaths than another.

(f) Finally, all issues listed above are not caused by class imbalance (the occurrence of one class with a higher frequency than another). In our story, for example, the two classes were perfectly balanced. Yet all these issues can worsen for imbalanced datasets[8].

But our story also points to a possible solution for all these issues. The 'metric' that ultimately proved to be relevant to the manager was the average net monetary gain obtained by using a candidate classifier. In the medical variation discussed in issue (d) above, it was the average life expectancy. In either case, such metric could have been easily calculated beforehand, upon gathering information about the average gains and losses of correct and incorrect classification, collected in the matrix (4) or (6), and combining these with statistics collected in the confusion matrix associated with the classifier. Denoting the former kind of matrix by (U_{ij}) and the confusion matrix by (C_{ij}), where i indexes the classifier outputs (rows) and j the true classes (columns), such a metric would have the formula

$$\sum_{i,j} U_{ij}\, C_{ij} \tag{7}$$

the sum extending to all matrix elements.

[8] Jeni et al. 2013; Zhu 2020.

In the present work, we argue that formula (7) is indeed the only acceptable metric for evaluating and comparing the performance of two or more classifiers, each with its own confusion matrix (C_{ij}) collected on relevant test data. The coefficients U_{ij}, called *utilities*, are problem-dependent. This formula is the *utility yield* of a classifier having confusion matrix (C_{ij}).

Our argument is based on *Decision Theory*, an overview of which is given in § 2.

The utility yield (7) is a linear combination of the confusion-matrix elements, with coefficients independent of the elements themselves. In § 3 we explore some properties of this formula and of the space of such metrics for binary classification. We also show that some common metrics such as precision, F_1-measure, Matthews correlation coefficient, balanced accuracy, and Fowlkes-Mallows index *cannot* be written as a linear combination of this kind. This impossibility has two consequences for such a metric. First, it means that the metric is always affected by some kind of cognitive bias. Second, there is *no* classification problem in which the metric correctly ranks the performance of all pairs of classifiers: using such a metric always leaves open the possibility that the evaluation is incorrect *a priori*. On the other hand, metrics such as accuracy, true-positive rate, true-negative rate can be written in the form (7). Consequently, each has a set of classification problems in which it correctly ranks the performance of *all* pairs of classifiers.

What happens if we are uncertain about the utilities appropriate to a classification problem? And what happens if the utilities are incorrectly assessed? We show in § 4 that uncertainty about utilities still leads to a metric of the form (7). We also show that an evaluation using incorrect utilities, even with relative errors as large as 20% of the maximal utility, still leads to a higher amount of correctly ranked classifiers than the use of any other popular metric.

Some remarks about the area under the curve of the receiver operating characteristic from the standpoint of our decision-theoretic approach are given in § 5.

We summarize and discuss our results in the final § 6.

2 Brief overview of decision theory

2.1 References

Here we give a brief overview of decision theory. We only focus on the notions relevant to the problem of evaluating classifiers, and simply state the rules of the theory. These rules are quite intuitive, but it must be remarked that they are constructed in order to be logically and mathematically self-consistent: see the following references. For a presentation of decision theory from the point of view of artificial intelligence and machine learning, see Russell & Norvig 2022 ch. 15. Simple introductions are given by Jeffrey 1965; North 1968; Raiffa 1970, and a discussion of its foundations and history by Steele & Stefánsson 2020. For more thorough expositions see Raiffa & Schlaifer 2000; Berger 1985; Savage 1972; and Sox et al. 2013; Hunink et al. 2014 for a medical perspective. See also Ramsey's 1926 insightful and charming pioneering discussion.

2.2 Decisions and classes

Decision theory makes a distinction between

- the possible situations we are uncertain about: in our case, the possible classes;
- the possible decisions we can make.

This distinction is important because it prevents the appearance of various cognitive biases[9] in evaluating the probabilities and frequencies of the possible situations on the one hand, and the values of our decisions on the other. Examples are the scarcity bias[10] "this class is rare, *therefore* its correct classification must lead to high gains", and plain wishful thinking: "this event leads to high gains, *therefore* it is more probable".

Often even the number of classes and the number of decisions differ. But in using machine-learning classifiers, one typically considers situations where the set of available decisions and the set of possible classes have some kind of natural correspondence and equal cardinality. In a 'cat vs dog' image classification, for example, the classes are 'cat' and 'dog', and the decisions could be 'put into folder Cats' vs 'put into folder

[9] Kahneman et al. 2008; Gilovich et al. 2009; Kahneman 2011. [10] Camerer & Kunreuther 1989; Kim & Markus 1999; Mittone & Savadori 2009.

Dogs'. In a medical application the classes could be 'ill' and 'healthy' and the decisions 'treat' vs 'dismiss'. As already mentioned, most of our discussions and examples focus for simplicity on binary classification.

2.3 Utilities and maximization of expected utility

To each decision we associate several *utilities*, depending on which of the possible classes is actually true. A utility may, for instance, equal a gain or loss in money, energy, number of customers, life expectancy, or quality of life, measured in appropriate units; or it may equal a combination of such quantities.

These utilities are collected into a *utility matrix* (U_{ij}), like the ones shown in formulae (4), (5), (6). The component U_{ij} is the utility of the decision corresponding to class i if class j is true, or simply the utility of class i conditional on class j.

In an individual classification instance, if we know which class is true, then the optimal decision is the one having maximal utility among those conditional on the true class. If, on the other hand, we are uncertain about which class is true, with probability p_j for class j such that $\sum_j p_j = 1$, then decision theory states that the optimal decision is the one having maximal *expected* utility $\bar{U}_i$, defined as the expected value of the utility of decision i with respect to the probabilities of the various classes:

$$\bar{U}_i \coloneqq \sum_j U_{ij}\, p_j \,. \tag{8}$$

In formulae, this principle of *maximization of expected utility* is

$$\text{choose class} \quad i^* = \arg\max_i \{\bar{U}_i\} \equiv \arg\max_i \left\{ \sum_j U_{ij}\, p_j \right\} . \tag{9}$$

A very important result in decision theory is that basic requirements of rational decision-making imply that there *must* be a set of utilities underlying the decisions of a rational agent, and the decisions must obey the principle of maximization of expected utility[11].

How are utilities determined? They are obviously problem-specific and cannot be given by the theory (which would otherwise be a model rather than a theory). Utilities can be obvious in decision problems

[11] Russell & Norvig 2022 § 15.2; von Neumann & Morgenstern 1955 chs 2–3.

involving gains or losses of measurable quantities such as money or energy (the utility of money is usually not equal to the amount of money, the relationship between the two being somewhat logarithmic[12]). In medical problems they can correspond to life expectancy and quality of life; see for example Sox et al. 2013 esp. ch. 8 and § 11.2.9 and Hunink et al. 2014 esp. ch. 4 on how such health factors are transformed into utilities.

The final utility of a single classification instance may depend, in some cases, on a sequence of further uncertain events and further decisions. In the story of § 0, for instance, the misclassification of a short-durability component as a long-durability one leads the final device to break only in a high fraction of cases, and in such cases the end customer requires a refund in a high fraction of subcases; the refunded amount may even depend on further circumstances. The negative utility $U_{01} = -335\,€$ in table (4) comes from a statistical average of the losses in all these possible end results. This is the topic of so-called decision networks or influence diagrams[13]. The decision-theory subfield of *utility theory* gives rules that guarantee the mutual consistency of a set of utilities in single decisions or decision networks. For simple introductions to utility theory see Russell & Norvig 2022 § 15.2, North 1968 pp. 201–205, and the references given at the beginning of the present section.

In the present work, we do not worry about such rules in order not to complicate the discussion: they should be approximately satisfied if the utilities of a problem have been carefully assessed.

3 Evaluation of classifiers from a decision-theoretic perspective

3.1 Admissible evaluation metrics for classification problems

Maximization of expected utility is the ground rule for rational decision making[14]. In the present work we focus on the stage where a large number of classifications have already been made by a classifier on a test dataset with N data. Denote by F_{ij} the number of instances in which the classifier chose class i and the true class was j. Then (F_{ij}) is the confusion matrix of the classifier on this particular test set. For all instances in

[12] e.g. North 1968 pp. 203–204; Raiffa 1970 ch. 4. [13] Besides the general references already given: Russell & Norvig 2022 § 15.5; Howard & Matheson 2005; for a step-by-step tutorial: Raiffa 1970. [14] We discuss and use it in our companion work Dyrland et al. 2022.

which the classifier chose class i and the true class was j, a utility U_{ij} is eventually gained. The total utility yielded by the classifier on the test set is therefore $\sum_{ij} U_{ij}\, F_{ij}$. Dividing by N we obtain the average utility per datum, which we call the *utility yield*; it can be written as

$$\boxed{\sum_{ij} U_{ij}\, C_{ij}} \tag{10}$$

where $C_{ij} \coloneqq F_{ij}/N$ is the relative frequency of choice i and true class j, and (C_{ij}) is the normalized confusion matrix.

The utility yield, formula (10)*, is therefore the natural metric to evaluate and compare the performance of classifiers on a test set for a classification problem characterized by the utility matrix* (U_{ij}).

Note how the utilities U_{ij} cannot depend on the frequencies F_{ij} or C_{ij}. If they did, it would mean that we had waited until *all* classification instances had been made in order to assess the value of each *single* instance. This would be a source of evaluation bias, such as the scarcity bias mentioned in § 2.2. It would, moreover, be an impossible procedure in contexts where the consequence of a single classification is manifest before the next classification is made.

If we modify the elements of a utility matrix by a common additive constant or by a common positive multiplicative constant,

$$U_{ij} \mapsto a\, U_{ij} + b \qquad a > 0\,, \tag{11}$$

then the final utilities yielded by a classifier with a particular confusion matrix are modified by the same constants. The ranking of any set of classifiers will therefore be the same. After all, an additive constant or a positive factor represent only changes in the zero or the measurement unit of our utility scale[15]. Such changes should not affect a decision problem. Indeed, the fact that they do not is another example of the logical consistency of decision theory.

3.2 Space of utility matrices for binary classification

Let us consider a problem of binary classification. It is characterized by a matrix of 2×2 utilities. We suppose that they are not all equal; the choice of class would be immaterial otherwise, and the classification problem

[15] cf. Russell & Norvig 2022 § 15.2.2.

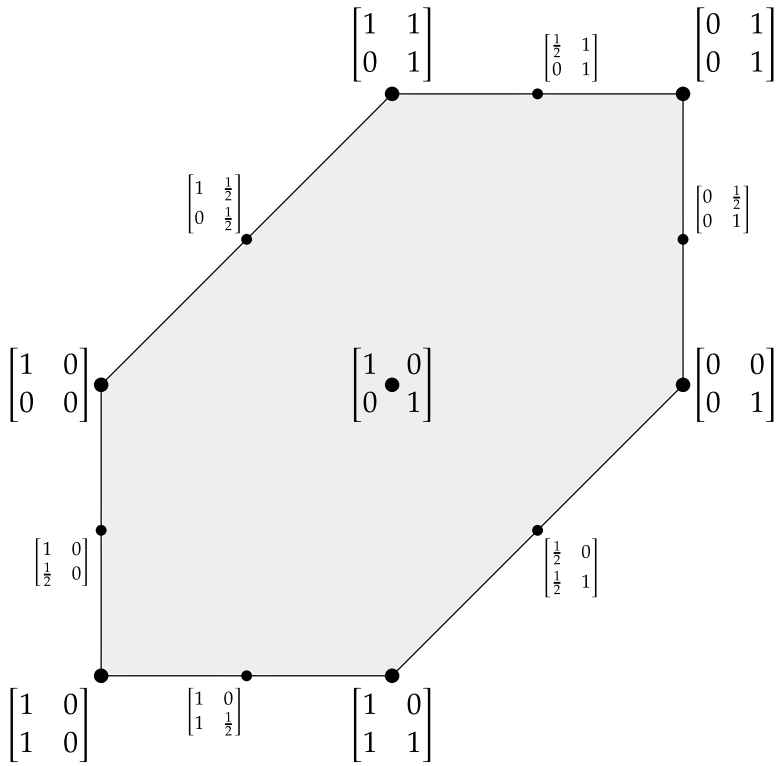


Figure 1 Space of utility matrices for binary classification.

trivial. We can use the freedom of choosing a zero and measurement unit to bring the utility matrix to a standard form. Let us choose them such that the maximum utility is 1 and the minimum utility is 0 (note that this value may still correspond to an actual monetary loss, for example). That is, we are effecting the transformation

$$U_{ij} \mapsto \frac{U_{ij} - \min(U_{ij})}{\max(U_{ij}) - \min(U_{ij})} \,. \tag{12}$$

With this convention, it is clear that we only have two degrees of freedom in choosing the utility matrix of a binary-classification problem. As a consequence, *the space of possible evaluation metrics for binary classifications is two-dimensional*. In order to evaluate candidate classifiers for a binary-classification problem, we must choose a point from this space.

We can represent this space as in fig. 1. The centre is the utility matrix with equal maximum utilities for correct classification and equal minimum utilities for incorrect classification; we shall see later that it corresponds to the use of accuracy as the evaluation metric. Moving to the left from the centre, the utility for correct classification of class 1 decreases with respect to class 0; vice versa moving to the right. Moving

upwards from the centre, the utility for misclassification of class 1 increases; moving downwards, the utility for misclassification of class 0 increases. We have excluded utility matrices in which misclassification has a higher utility than correct classification (although they may occur in some situations); they would appear in the missing upper-left and lower-right corners. Fixing (x, y) axes through the centre of the set, a utility matrix has coordinates

$$\begin{bmatrix} 1 - x\,\delta(x > 0) & y\,\delta(y > 0) \\ -y\,\delta(y < 0) & 1 + x\,\delta(x < 0) \end{bmatrix} . \tag{13}$$

Note that this representation is *not* meant to reflect any convex or metric properties, however. No metric or distance is defined in the space of utility matrices. Convex combination is defined if we drop the normalization (12) but it is not correctly reflected in the representation of fig. 1.

3.3 Relationship with common metrics

In § 3.1 we found that the most general evaluation metric according to decision theory must be a linear combination of the confusion-matrix elements. The coefficients of this linear combination cannot depend on the confusion-matrix elements themselves, because such a dependence would reflect some sort of cognitive bias. Which common popular metrics adhere to this mathematical form? We want to answer this question in the binary-classification case while giving as much allowance as possible in the typical context in which popular metrics are used.

Consider the case in which we are comparing several classifiers *on the same test set*. The number of data N and the relative frequencies f_0, f_1 with which the two classes '0', '1' occur in the test set are fixed and constant for all classifiers under evaluation.

A classifier yields a normalized confusion matrix (C_{ij}) which we write in the format

$$\begin{array}{cc} & \textit{true class} \\ & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} \textit{classifier} \\ \textit{output} \end{array} \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} \end{array} .$$

Owing to the constraints $C_{00} + C_{10} \equiv f_0$ and $C_{01} + C_{11} \equiv f_1$ we can always make two elements of the confusion matrix appear or disappear

from any formula, replacing them with expressions involving the remaining two elements and the class frequencies. To avoid ambiguities in interpreting the functional form of mathematical formulae, let us agree to always express them in terms of C_{00} and C_{11} only, making the replacements $C_{10} = f_0 - C_{00}$, $C_{01} = f_1 - C_{11}$ wherever necessary.

Recall that, given a utility matrix, we can always modify its elements by a common positive multiplicative constant a and by a common additive constant b, eq. (11), because such a modification corresponds to a change of unit and zero of the utility scale. With such a modification the evaluation metric (10) takes the equivalent form

$$a \sum_{ij} U_{ij}\, C_{ij} + b \tag{14}$$

because $\sum_{ij} C_{ij} \equiv 1$. Writing the sum explicitly and rewriting the elements C_{10}, C_{01} in terms of C_{00}, C_{11} as discussed above, this formula becomes

$$a\,(U_{00} - U_{10})\, C_{00} \;+\; a\,(U_{11} - U_{01})\, C_{11} \;+\; a\, f_0\, U_{10} + a\, f_1\, U_{01} + b\,. \tag{15}$$

Since in the present context N, f_0, f_1 are constants, we are free to construct the arbitrary constants $a > 0$ and b from them in any way we please:

$$a = a(N, f_0, f_1) > 0\,, \qquad b = b(N, f_0, f_1)\,. \tag{16}$$

We can also use this freedom to include the term $a\, f_0\, U_{10} + a\, f_1\, U_{01}$ into b in the formula above. We conclude that *an evaluation metric for binary classification complies with decision theory if and only if it can be written in the general form*

$$a(N, f_0, f_1)\, X\, C_{00} + a(N, f_0, f_1)\, Y\, C_{11} + b(N, f_0, f_1) \tag{17}$$

where X, Y are constants that do not depend on $C_{00}, C_{11}, N, f_0, f_1$; and $a(\,\cdot\,) > 0$, $b(\,\cdot\,)$ are arbitrary functions of N, f_0, f_1 only.

A monotonic function (such as an exponential) of such form is also admissible if we only require a comparison score to rank several classifiers from best to worst.

Let us examine some common evaluation metrics for binary classification from this point of view. We write their formulae in terms of C_{00}, C_{11}.

The following metrics are particular instances of formula (17):

- ✓ *Accuracy:* $C_{00} + C_{11}$. We have $a = 1$, $X = Y = 1$, $b = 0$. Indeed it corresponds to the utility yield based on the identity utility matrix $(U_{ij}) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ (or equivalently a utility matrix that assigns the same utility to the correct classification of any class, and the same, lower utility to the misclassification of any class).
- ✓ *True-positive rate (recall):* C_{00}/f_0. Here $a = 1/f_0$, $X = 1$, $Y = 0$, $b = 0$. It corresponds to using the utility matrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$.
- ✓ *True-negative rate (specificity):* C_{11}/f_1. Here $a = 1/f_1$, $X = 0$, $Y = 1$, $b = 0$. It corresponds to using the utility matrix $\left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right]$.

The following metrics instead *cannot* be written in the form (17), nor as monotonic functions of that form:

- ✗ *Precision:* $C_{00}/(C_{00} - C_{11} + f_1)$. Non-linear in C_{00}, C_{11}.
- ✗ *F_1-measure:* $2C_{00}/(C_{00} - C_{11} + 1)$. Non-linear in C_{00}, C_{11}. The same is true for the more general F_β-measures.
- ✗ *Matthews correlation coefficient:* $\frac{f_1\, C_{00} + f_0\, C_{11}}{\sqrt{f_0\; f_1\; (f_1 + C_{00} - C_{11})\; (f_0 + C_{11} - C_{00})}}$. Non-linear in C_{00}, C_{11}.
- ✗ *Fowlkes-Mallows index:* $C_{00}/\sqrt{f_0\,(f_1 + C_{00} - C_{11})}$. Non-linear in C_{00}, C_{11}.
- ✗ *Balanced accuracy:* $C_{00}/(2f_0) + C_{11}/(2f_1)$. Despite being linear in C_{00}, C_{11} and an average of two metrics (true-positive and true-negative rate) that are instances of formula (17), it is not an instance of that formula, because the two averaged metrics involve different $a(\,\cdot\,)$ functions.

We see that many popular evaluation metrics do not comply with the principles of decision theory. Any such metric suffers from two problems.

First, as discussed in § 2, the metric involves an interdependence of utilities and classification frequencies, which implies some form of cognitive bias[16].

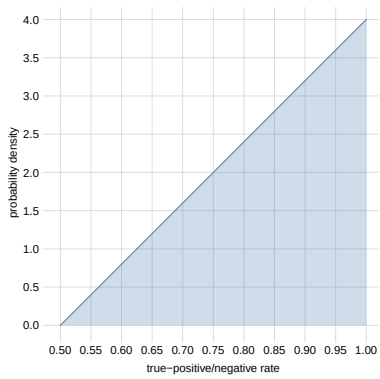
Second, the ranking of confusion matrices yielded by the metric does not fully agree with that yielded by any utility matrix – a full agreement would otherwise imply that the metric could be written in the form (17). Some confusion matrices must therefore be incorrectly ranked. Since any rational classification problem is characterized by some underlying

[16] Hand & Christen 2018 discuss such biases regarding the F_1-measure.

utility matrix, this means that the incompliant metric will always lead to some wrong evaluations. By contrast, compliant metrics such as the accuracy give completely correct rankings for all pairs of confusion matrices in specific sets of classification problems.

The second phenomenon is illustrated in the plots of figs 2–3. Each blue dot in a plot represents a hypothetical confusion matrix obtained from a test dataset in a binary classification. The dot's coordinates are the utility yield of that confusion matrix according to a particular utility matrix underlying the classification problem, and the score of the confusion matrix according to another metric. The underlying utility matrix is $\left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$ for all plots in the left column, and $\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$ for all plots in the right column. The other metrics considered, one for each row of plots, are accuracy, true-positive rate (recall, class 0 being 'positive'), F_1-measure, Matthews correlation coefficient.

The confusion matrices are selected by first fixing a proportion of classes in the dataset, which is 50%/50% (balanced dataset) for all plots in fig. 2 and 90%/10% (imbalanced dataset) for all plots in fig. 3. Then a true-positive rate and a true-negative rate are independently selected from the range $[1/2, 1]$, with a probability linearly increasing in the rate (median of 0.85, lower and upper quartiles at 0.75 and 0.93; see side plot). These confusion matrices therefore represent the classification statistics produced by classifiers that tend to have good performance – as is clear from the fact that the points tend to accumulate on the upper-right corners of the plots.



We see that the accuracy (first-row plots) always gives correct relative evaluations of all confusion matrices when the underlying utility matrix is equivalent to $\left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$ (left column): the y-coordinate is a monotonically increasing function – in fact a linear function – of the x-coordinate. Accuracy is indeed the utility yield corresponding to the identity utility matrix. The true-positive rate (second-row plots) always gives correct relative evaluations (provided the test set is the same) when the underlying utility matrix is equivalent to $\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$ (right column).

On the other hand, if any of these two metrics is used for a problem having a different underlying utility matrix, then there is no deterministic relationship between the metric's score and the actual utility yield. In

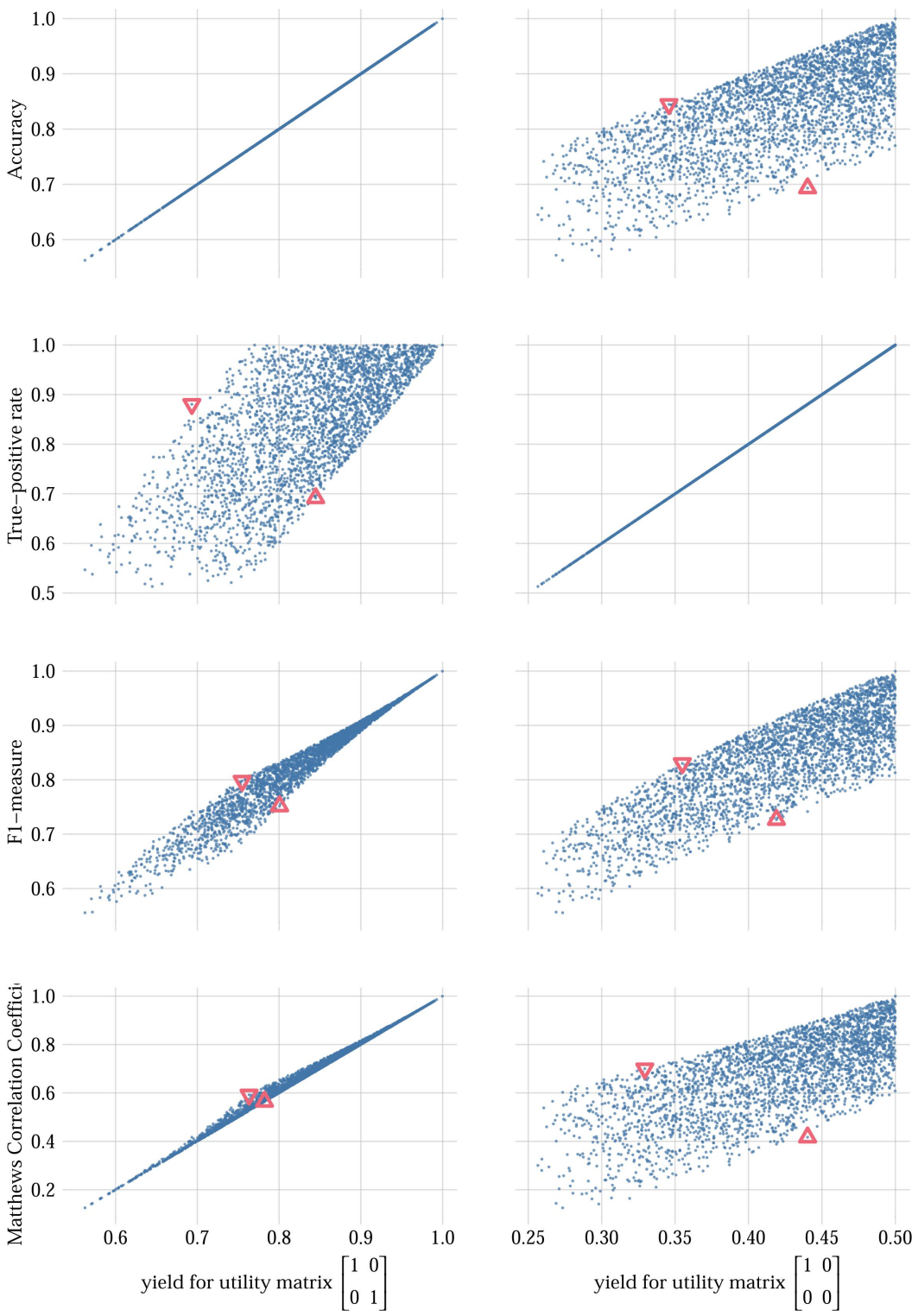


Figure 2 Relationship between various evaluation metrics and actual utility yields for two different binary-classification problems with underlying utility matrices $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ (left column) and $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ (right column). All confusion matrices (blue dots) are obtained from a dataset with 50%/50% class balance. Pairs of red triangles in a plot show two confusion matrices that are wrongly ranked by the metric (y-axis) with respect to the actual utility yield (x-axis). Clearly, there can even be three or more confusion matrices ranked in completely reverse order by the metric. The accuracy yields correct evaluations the classification problem on the left column; and the true-positive rate, for the one on the right.

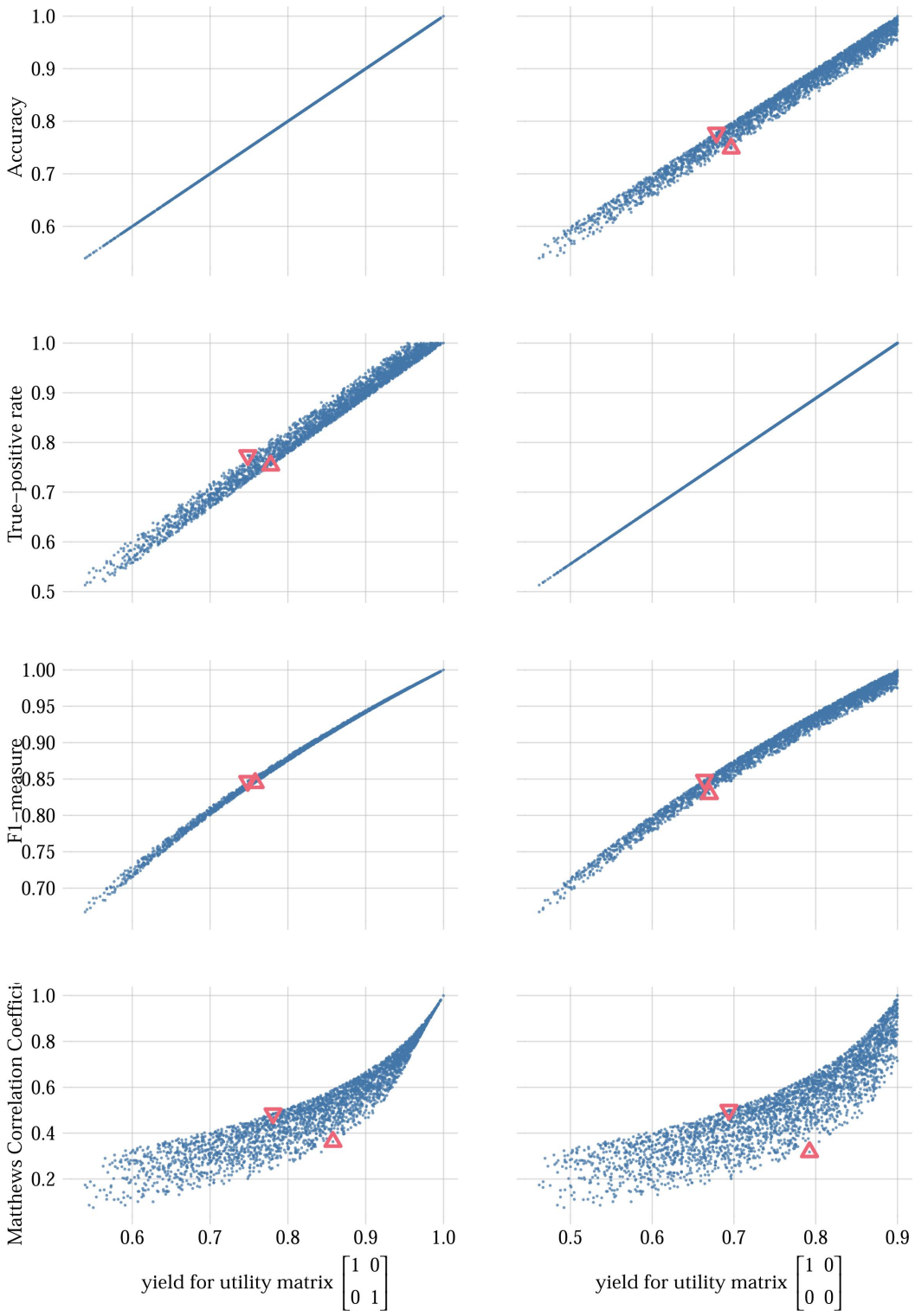


Figure 3 As for fig. 2 but for confusion matrices obtained from an imbalanced dataset with 90% occurrence of class 0 ('positive') and 10% of class 1 ('negative').

this case it is always possible to find two or more confusion matrices for which the metric gives completely reversed evaluations with respect to the actual utility yield. In other words, the confusion matrix – and associated algorithm – which is worst according to the true utility, is ranked best by the metric; and vice versa. Pairs of red triangular shapes in a plot are examples of such confusion matrices wrongly ranked by the y-axis metric.

Metrics such as accuracy and true-positive rate, complying with formula (17), thus require us to rely on evaluation *luck* only when they are used in the wrong classification problem.

The plots for the F_1-measure (third-row plots) and Matthews correlation coefficient (fourth-row plots) show that these two metrics do not have any functional relationship with the actual utility yield. It is again always possible to find two or more confusion matrices for which either metric gives completely reversed evaluations with respect to the actual utility yield. But for these two metrics, unlike accuracy and true-positive rate, cases of incorrect evaluation will *always* occur in *every* classification problem.

Metrics such as F_1-measure and Matthews correlation coefficient, not complying with formula (17), thus *always require us to rely on luck in our evaluations*. There are no classification problems for which these metrics lead to always correct evaluations.

A metric non-compliant with decision theory can lead to a large number of correct results for some classification problems and test sets. The bottom-left plot of fig. 2, for instance, shows that the Matthews correlation coefficient is almost a monotonically increasing deterministic function of the utility yield when the underlying utility matrix is the identity and the dataset is balanced (but it is not when the underlying utility matrix is $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ or the dataset is imbalanced; see corresponding plots). Such an occasional partial agreement is useless, however. Knowledge of the utility matrix is a prerequisite for relying on such partial agreement– but given such knowledge we can directly use the actual utility yield instead, which has an exact agreement and is easier to compute.

4 Unknown or incorrect utilities

So far, we have argued that the natural evaluation metric for a classifier is the utility yield of its confusion matrix, according to the utilities

underlying the classification problem of interest. We have also argued that many popular metrics, those not complying with formula (17), must always a priori lead to instances of incorrect evaluation. Our arguments are based on the principles of decision theory.

Several interrelated questions spring from our arguments, though:

- What to do when we are uncertain about the utilities underlying a classification problem?
- What happens if the utilities we use are actually wrong, that is, not the true ones underlying the problem?
- How often do uncompliant metrics such as F_1-measure or Matthews correlation coefficient lead to incorrect results, on average?

In fact, if a small error in the assessment of the utilities led to a large number of wrong evaluations, while incompliant metrics led to a small number of wrong evaluations on average, then all the rigorousness of decision-theoretic metrics would be useless in practice, and incompliant metrics would be best for real applications.

This is not the case, however. We now discuss how to deal with uncertainty about the utilities and present an important result: Using wrong utilities, even with relative errors almost as large as 20% of the maximum utility, still leads to fewer incorrect relative evaluations on average than using many currently popular metrics.

4.1 Unknown utilities; average performance on several classification problems

Dealing with unknown utilities is straightforward. Suppose we are uncertain whether the utility matrix appropriate to a classification problem is $\boldsymbol{U}^{(1)} \equiv \bigl(U^{(1)}_{ij}\bigr)$, or $\boldsymbol{U}^{(2)}$, or $\boldsymbol{U}^{(3)}$, and so on, where the number of alternatives can even be infinite or continuous. Each alternative $\boldsymbol{U}^{(a)}$ has a probability q_a, or probability density $q(a)\,\mathrm{d}a$ in the continuous case. Then *for this classification problem, we should use the expected utility matrix*

$$\hat{\boldsymbol{U}} := q_1\,\boldsymbol{U}^{(1)} + q_2\,\boldsymbol{U}^{(2)} + q_3\,\boldsymbol{U}^{(3)} + \cdots \tag{18}$$

or $\hat{\boldsymbol{U}} := \int q(a)\,\boldsymbol{U}^{(a)}\,\mathrm{d}a$ *in the continuous case.*

We only give a sketch of the proof of this intuitive result[17]. If we are uncertain about the utility matrix, then we have a double decision

[17] see e.g. Raiffa 1970 esp. ch. 3.

problem: choosing the optimal utility and choosing the optimal class. If the true utility matrix is, for instance, $\boldsymbol{U}^{(2)} \equiv \big(U^{(2)}_{ij}\big)$, and the true class is class 0, then choosing class 1 would yield a utility $U^{(2)}_{10}$, choosing class 0 would yield a utility $U^{(2)}_{00}$, and so on. Our double decision problem is thus characterized by a rectangular utility matrix that is the row-concatenation of the utility matrices $\boldsymbol{U}^{(a)}$. We make the realistic judgement that the probabilities q_a of the utility matrices and the probabilities p_j of the classes are independent, so that $q_a \cdot p_j$ is the probability that the true utility matrix is $\boldsymbol{U}^{(a)}$ and the true class is j. The principle of maximum expected utility, § 2.3 eq. (9), then leads to the maximization of the expected utilities

$$\bar{U}_i \coloneqq \sum_{j,a} U^{(a)}_{ij}\, q_a \cdot p_j \equiv \sum_j \underbrace{\left[\sum_a q_a\, U^{(a)}_{ij}\right]}_{\hat{\boldsymbol{U}}} p_j \tag{19}$$

in which the expected utility matrix (18) appears as an 'effective' utility matrix to be used for the class-decision problem alone.

If our uncertainty is symmetric with respect to the utilities conditional on the different classes – for instance, our uncertainty about the utilities conditional on class 0 is the same as on class 1 – then the expected utility matrix is equivalent to the identity matrix. The utility yield is in this case equal to the accuracy. The accuracy is therefore the natural evaluation metric to use if we are in a complete state of uncertainty regarding the underlying utilities. This fact is indeed reflected in some results discussed in § 4.2.

For binary classification the set of possible utility matrices can be represented as in fig. 1, as discussed in § 3.2. Our uncertainty about the true underlying utility matrix corresponds to a discrete or continuous distribution of probability over this set. Note, however, that the expected utility matrix (18) does *not* correspond to the mass-centre of the distribution, because of the peculiar coordinate system used in that figure. The actual mass-centre is obtained by representing the set of utility matrices as a two-dimensional surface (a tetrahedron) in three-dimensional space. For brevity we do not discuss this representation in the present work.

The procedure of averaging utilities, formula (18), also applies if we want to evaluate how a classifier performs on average on several

classification problems, which differ in their utility matrices. Again, what we need to use is the average of their utility matrices.

4.2 Consequences of wrong utility assessments and comparison with common metrics

It may happen that our assessment of the utility matrix of a classification problem is incorrect, especially if it has been made on semi-quantitative grounds owing to a lack of information. Then our comparative evaluations of classifiers may also end up being incorrect. What is the probability of an incorrect comparative evaluation, on average, in such cases? and how does it depend on the amount of error in the utilities? Is it higher than the probability of incorrect evaluation by other metrics?

A precise answer to these questions is extremely difficult if not impossible because to define 'on average' we would need to conduct a survey of classification problems of any kind, collecting statistics about their underlying utility matrices, about the confusion matrices of candidate classification algorithms for their solution, and about the errors committed in assessing utilities. We try to give a cursory answer to the questions above for the binary-classification case, based on the following assumptions and judgements:

(i) Two possible distributions of true utility matrices on the set of fig. 1 (in that coordinate system): 1. a uniform distribution; 2. a bivariate (truncated) gaussian distribution centred on the identity matrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and with standard deviation $1/3$ in the x and y coordinates of eq. (13), illustrated in fig. 4.

(ii) A distribution of confusion matrices for which the fraction of one class is uniformly distributed in $[0, 1]$, and the true-positive and true-negative rates are independently distributed in $[0.5, 1]$ with linearly increasing probabilities (median of 0.85, lower and upper quartiles at 0.75 and 0.93; see side plot on p. 16). This means that we consider problems with highly imbalanced data to be as common as problems with balanced data (a realistic assumption, according to our experience), and candidate classifiers to be generally good.

(iii) A truncated gaussian distribution of error around each true utility-matrix element, centred on the true utility value. We consider standard deviations ranging from 0 to 0.3. The gaussian must be truncated because each true utility has a value between 0 and 1, and

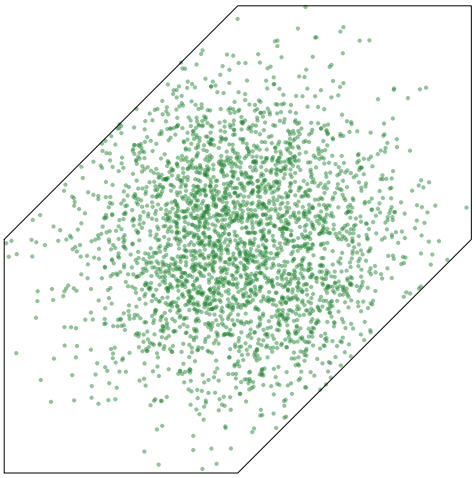


Figure 4 Truncated gaussian distribution in the space of utility matrices of fig. 1, described in item (i).

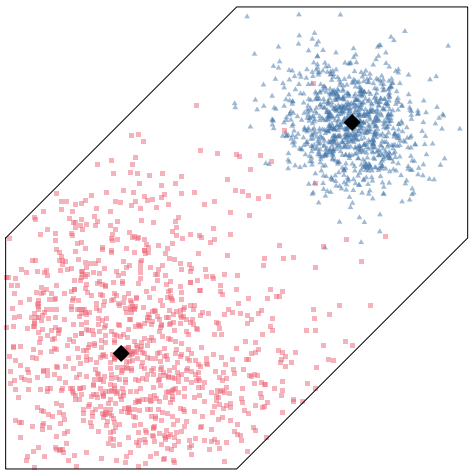


Figure 5 Extents of errors having standard deviations 0.1 (blue triangles) and 0.2 (red squares), around the utility matrices $\begin{bmatrix} 0.5 & 0.5 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \end{bmatrix}$ (black diamonds).

because we require the utilities of correct classifications to be larger than those of incorrect ones. Figure 5 illustrates the extent of such an error in the space of utility matrices, for standard deviations equal to 0.1 (blue triangles) and 0.2 (red squares).

Under these assumptions, we calculate how often a pair of classifiers, having two confusion matrices with the same class proportions, is evaluated in reverse order, with respect to their true utility yield, when an incorrect utility matrix or another metric is used for the evaluation. This calculation is an integration problem that we solve by Monte Carlo sampling. The procedure is intuitive:

1. Select a 'true' utility matrix according to the distribution (i).
2. Select errors around the elements of the true utility matrix, according to the distribution (iii), and add them to it.
3. Select a class proportion and then two confusion matrices having that class proportion (the class proportion must be the same since the matrices are obtained from the same data), according to the distributions (ii).
4. Calculate the signed difference between the true utility yield of the second confusion matrix and that of the first confusion matrix, using the true utility from step 1. If this difference is positive, then

the second confusion matrix has higher utility than the first; if negative, then the first confusion matrix has higher utility than the second.

5. a. Consider several metrics (precision, Matthews correlation coefficient, and so on). For each, calculate the signed difference between the score it gives to the second confusion matrix, and the score it gives to the first.
   b. Consider the erroneous utility matrix from step 2. Calculate the signed difference between the utility yield of the second confusion matrix and that of the first confusion matrix, using this erroneous utility matrix.

   In either case, a positive difference means that the second confusion matrix is ranked 'best' and the second 'worst', and vice versa for a negative difference.

6. Now go through the signed differences obtained in step 5, and compare them, in turn, with the signed difference obtained in step 4. If the difference from step 5 has opposite sign to that of step 4, then the two confusion matrices are oppositely and incorrectly ranked by the corresponding metric or by the erroneous utility matrix.

The results of this sampling procedure for the case of uniform distribution of true utility matrices, several metrics, and utilities affected by errors with 0.1 standard deviation, are shown in fig. 6. Each point represents a pair of confusion matrices (step 3); its coordinates are the true utility yield and either the score given by a metric or (last plot) the yield according to the incorrect utility matrix. The red or yellow triangular points in the II and IV quadrants (discordant signs) are incorrectly ranked pairs. The percentages of incorrect rankings are calculated from 10^6 samples, giving slightly more than one decimal significant digit; fewer samples are shown in the plots.

The plots are displayed in order (left-right, top-bottom) of decreasing percentages of incorrect rankings. The accuracy metric proves to be the best among the ones considered, leading to 8.7% incorrect pairwise rankings. But we see that a utility matrix affected by gaussian errors with 0.1 standard deviation is even better, yielding 4% incorrect pairwise rankings.

The dependence of the fraction of incorrect rankings on the standard deviation of the error affecting the utilities is shown in the plots of fig. 7,

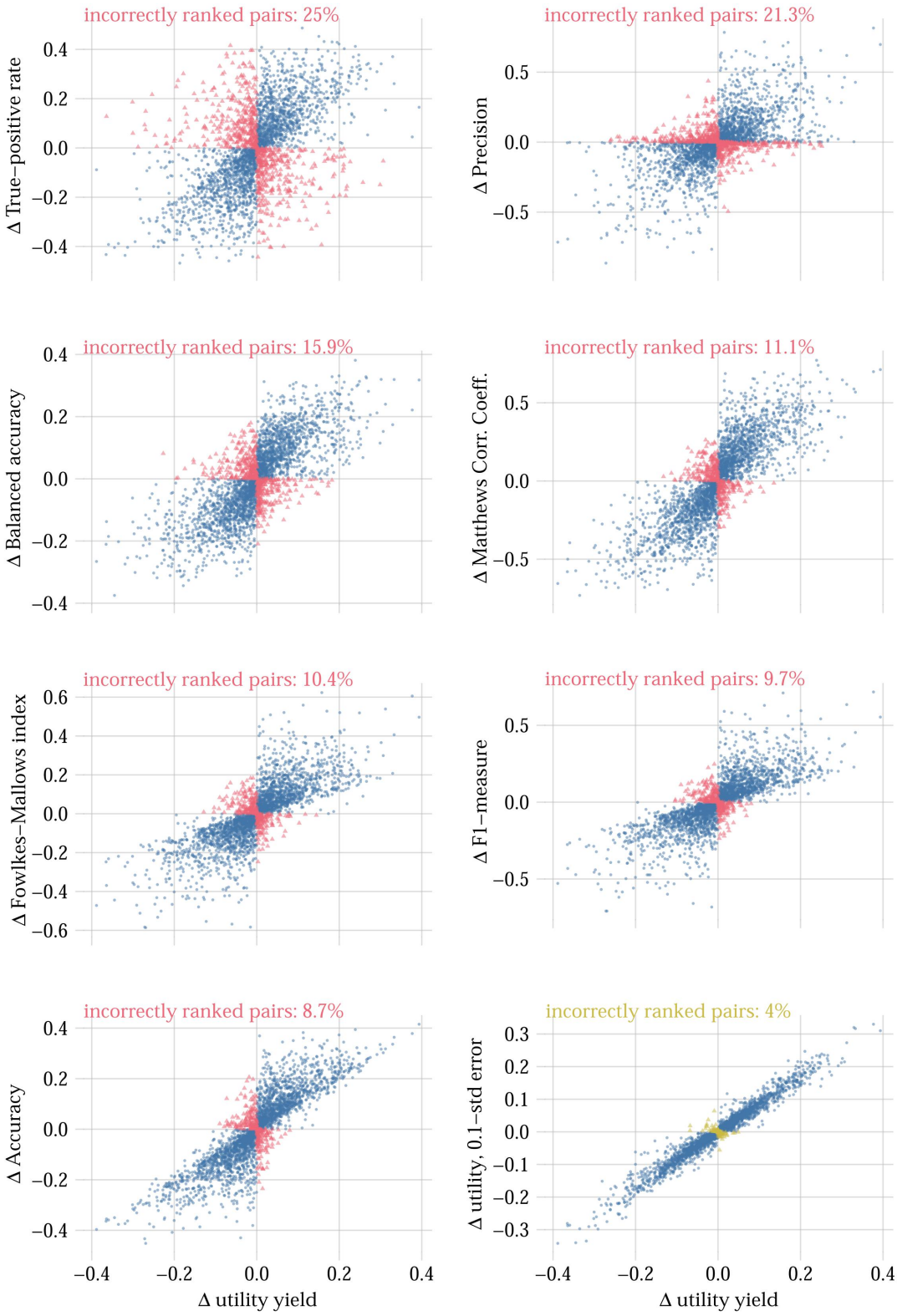


Figure 6 Relationship between difference in utility yields according to a 'true' utility matrix, and difference in scores according to other metrics including an incorrectly assessed utility matrix (error with 0.1 standard deviation). Points landing in the II or IV quadrants represent pairs of confusion matrices that were wrongly compared.

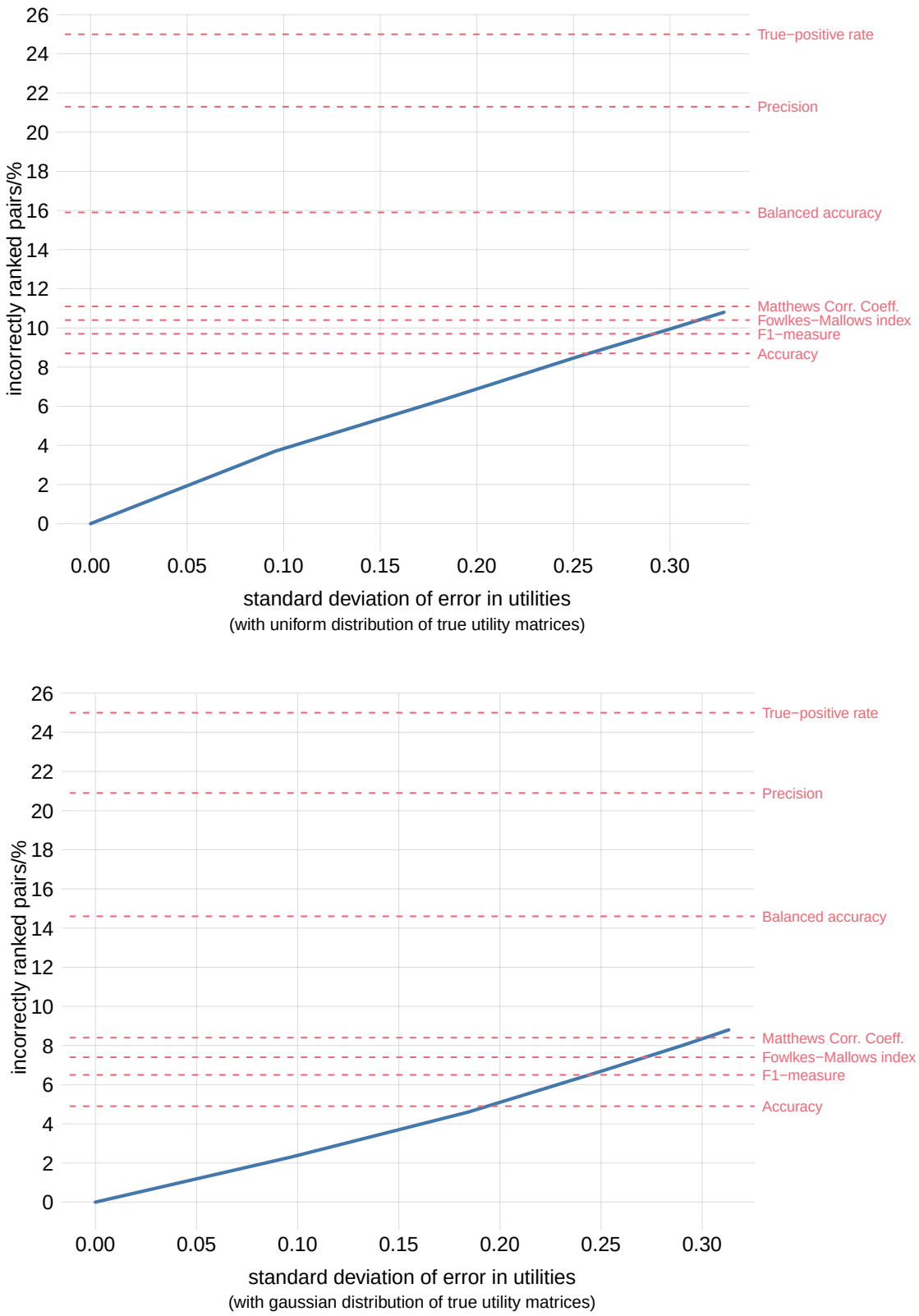


Figure 7 Dependence of the proportion of incorrectly ranked pairs of confusion matrices, on the standard deviation of the assessment error on the utilities. Top plot: case with uniform distribution of true utility matrices. Bottom plot: case with gaussian distribution of true utility matrices, as in fig. 4.

for the case of uniform distribution (top plot) and gaussian distribution (bottom plot) of true utility matrices. It is approximately linear. The plots also report the fractions of incorrect rankings for the other metrics. We see that evaluations based on a utility matrix affected by errors with standard deviation up to 0.15 or even 0.25 are still more reliable than evaluations based on the other reported metrics. This is a remarkable fact, considering that errors with such standard deviations are quite large, as was shown in fig. 5.

A utility error with standard deviations around 0.25 covers the whole space of utility matrices almost uniformly (cf fig. 5). Such a large error means that we are almost completely uncertain about the utilities to start with. It therefore makes sense that the accuracy, equivalent to using the identity utility matrix, becomes a more reliable metric when this error level is reached: as we saw in § 4.1, the identity utility matrix is the natural one to use in a state of complete uncertainty about the utilities. This result is just another example of the internal consistency of Decision Theory.

5 What about the area under the curve of the receiver operating characteristic?

Another very common metric for evaluating binary classifiers is the Area Under the Curve of the Receiver Operating Characteristic, or 'area under the curve' for short. This metric can only be used for particular classifying algorithms, and its meaning is different from that of the metrics reviewed so far. For these reasons, we leave a full discussion of it to future works and only offer a couple of remarks here.

The area under the curve can only be computed for classifiers that output a continuous variable rather than a class. A threshold for this variable determines whether its value predicts one class or the other. Different choices of threshold lead to different pairs of false-positive rate f (which is 1 − true-negative rate) and true-positive rate t in a given test set. These pairs can be plotted as a curve $f \mapsto t(f)$ on a graph with corresponding axes. Given the proportion of classes in the test set, every point on such curve corresponds to a possible confusion matrix $C_{ij}(f)$ that the classifier can produce depending on the threshold chosen. The area subtended by such curve is a weighted average of true-positive rates with a peculiar choice of weights; the weights are uniform as a function

of the false-positive rate, but generally not uniform as a function of the threshold, for example. The meaning and proper use of the receiver operating characteristic are discussed in a classic by Metz 1978, see especially p. 290.

From the standpoint of decision theory, two remarks can be made[18]. First, according to the principle of maximum expected utility, § 2.3, we should choose a threshold and corresponding false-positive rate f^* such as to maximize the utility yield, given by eq. (10):

$$\text{choose} \quad f^* = \arg\max_f \left\{ \sum_{i,j=0}^{1} U_{ij}\, C_{ij}(f) \right\} . \tag{20}$$

Any other values of f and of the threshold are irrelevant. Averages over f values are therefore irrelevant as well. Second, suppose our goal is to evaluate the average performance over several possible classification problems. In that case, the quantities to be averaged are the utility matrices of those classification problems, as discussed in § 4.1, yielding a unique expected utility matrix. Once this is computed, we go back to a single choice of f according to our first remark.

Owing to these issues, the area under the curve suffers from the same problems as the non-compliant metrics discussed in § 3.3: in every classification problem, it always leads to cases of incorrect evaluation.

A correct use of the receiver-operating-characteristic curve $t(f)$ can be made, however. It is explained in Metz 1978 section *Cost/Benefit Analysis* p. 295, and in Sox et al. 2013 § 5.7.4 (curiously Sox et al. also mention the generally erroneous criterion of the area under the curve).

Denote the proportion of class 0 (positive) in the test set by B. The confusion matrix as a function of f is then

$$\begin{bmatrix} C_{00}(f) & C_{01}(f) \\ C_{10}(f) & C_{11}(f) \end{bmatrix} = \begin{bmatrix} B\, t(f) & (1-B)\, f \\ B\, [1-t(f)] & (1-B)\,(1-f) \end{bmatrix} . \tag{21}$$

The sum in formula (20) above can then be explicitly written, rearranging some terms,

$$\sum_{i,j=0}^{1} U_{ij}\, C_{ij}(f) \equiv (U_{00}-U_{10})\, B\, t(f) - (U_{11}-U_{01})\,(1-B)\, f + U_{10}\, B \; + U_{11}\,(1-B) . \tag{22}$$

[18] similar points are made by Baker & Pinsky 2001; Lobo et al. 2008.

The principle of maximum expected utility (20) is then equivalent to the following condition, obtained using the explicit sum above but dropping the constant term on the second line for simplicity:

$$\text{choose} \quad f^* = \arg\max_f \bigl\{(U_{00} - U_{10})\, B\, t(f) - (U_{11} - U_{01})\,(1 - B)\, f\bigr\}\,. \tag{23}$$

The function in braces is monotonically increasing because $t(f)$ is (we assume, as always, that the utility of correct classification of a class is higher than that of misclassification, so $U_{00} - U_{10} \geqslant 0$ and $U_{11} - U_{01} \geqslant 0$). Its maximum can thus be found by setting its derivative to zero:

$$\text{choose } f^* \text{ such that} \quad t'(f^*) = \frac{(U_{11} - U_{01})\,(1 - B)}{(U_{00} - U_{10})\, B}\,. \tag{24}$$

If we have several classifiers, each with its own curve $t(f)$, then *the best is*

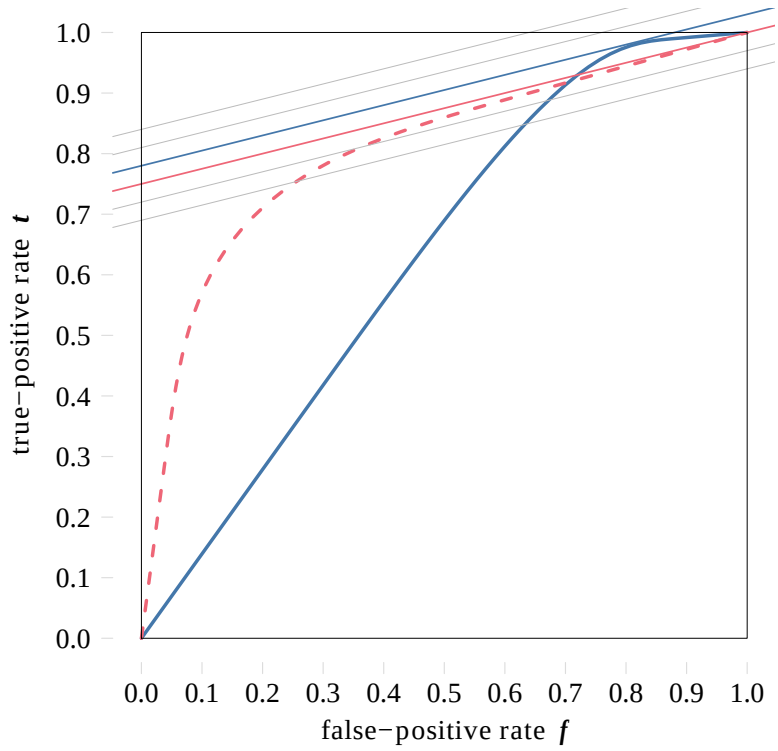


Figure 8 Receiver-operating-characteristic curves of two classifiers. The red dashed curve clearly subtends a larger area than the blue solid curve. Yet the classifier with the latter curve yields a higher utility, because it touches the family of parallel lines, eq. (25), at a higher point. This example arises for a utility matrix equal to $\left[\begin{smallmatrix} 4 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and a test set with $B = 0.5$ (balanced), or for a utility matrix equal to $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and a test set with $B = 0.8$.

the one tangent to the line

$$t = \frac{(U_{11} - U_{01})\,(1 - B)}{(U_{00} - U_{10})\,B} f + \text{const.} \tag{25}$$

that has the highest intercept.

From this criterion it can be seen geometrically that if a classifier has its curve $t(f)$ *completely above* the curve of another classifier, then it must have a higher utility yield. But nothing, in general, can be said if the curves of the two classifiers cross. It is the tangent of a receiver-operating-characteristic curve that matters, not its subtended area. Figure 8 shows an example of this.

6 Summary and discussion

The evaluation and ranking of classification algorithms is a critical stage in their development and deployment. Without such evaluation we cannot even say whether an algorithm is better than another, or whether a set of parameter values for a specific algorithm is better than another set.

And yet, at present, we have not an evaluation theory but only an evaluation folklore: different procedures, proposed only out of intuition and of analysis of special cases, with fuzzy criteria to decide which should be used, and without rigorous theoretical foundations that should guarantee uniqueness and universality properties and absence of biases. We believe that some of the surprising failures of machine learning *in actual applications*[19] come not only from biases in the choice of test datasets and other similar biases but also from the use of wrong evaluation metrics in the development stage.

In the present work, we have argued that theoretical foundations for the evaluation process are available in *Decision Theory*. Its main notions and principle – utilities and their maximization – are very intuitive, as shown (we hope) by the introductory story.

These are the main results of the application of decision theory to the evaluation of classifiers:

- The evaluation metric must depend on the specific classification problem.

[19] see e.g. Varoquaux & Cheplygina 2022.

• Such metric is completely defined by n^2 parameters, called utilities, collected in a utility matrix; n is the number of classes. Two parameters are arbitrary and represent a zero and measurement unit of the utility scale. In the binary-classification case, this means that we have a two-dimensional set of possible metrics.

• The score of a classifier on a test set is simply given by its utility yield: the grand sum of the products of the elements of the utility matrix and the confusion matrix of the classifier. It is a simple linear expression in the confusion-matrix elements.

• A utility matrix, obtained from an average, is also used when we are uncertain about the utilities underlying a classification problem or when we want to consider the average performance over several classification problems.

• Some popular metrics such as precision, balanced accuracy, Matthews correlation coefficient, Fowlkes-Mallows index, F_1-measure, and area under the receiver-operating-characteristic curve do not comply with decision theory. As a consequence, they are affected by cognitive biases and always lead to some erroneous comparative evaluations of classifiers in every classification problem, even when all utilities and frequencies are correctly assessed.

• Using a utility matrix with incorrectly assessed utilities still leads, on average, to fewer wrong comparative evaluations than using other popular metrics.

We believe that the decision-theoretic evaluation of classifiers also has remarkable advantages:

First, it translates the fuzzy problem "which of the numerous scores should I rely on?" into a more structured, thus easier to confront, one: to assess, at least semi-quantitatively, how many times more valuable, desirable, or useful is the correct classification of a class than its incorrect classification, than the correct classification of another class, and so on. Such utilities usually have a more immediate, problem-dependent interpretation than other metrics.

Second, it leads to a mathematically simple, computationally convenient metric: a linear combination of confusion-matrix elements – no need for non-linear functions or integration of curves.

Third, the principles of the underlying theory guide us if we have to face new peculiar problems. Imagine, for instance, a classification

problem where we cannot say, in general, whether true positives are more important than true negatives and so on, because such valuation can *vary from one tested item to another*. Decision theory, in this case, requires an item-wise assessment of utilities, and still provides an item-wise score, which can be accumulated across items to obtain a total evaluation score for the performance of candidate classifiers.

The theory, remarks, and results of the present work generalize beyond classification: to regression and more complex classification-like problems such as image segmentation, with important applications in medicine[20]. It would be interesting to examine whether popular metrics in the latter field, such as Dice score[21] and Hausdorff distance[22], comply with decision-theoretic principles, and which alternatives could be used otherwise.

In a companion work[23] we apply the general ideas presented here to improve the performance of machine-learning classifiers.

Author contributions

All authors have contributed equally to the present work.

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.

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, Python, Inkscape, LibreOffice, Sci-Hub for making a free and impartial scientific exchange possible.

[20] Lundervold & Lundervold 2019. [21] Dice 1945; Fleiss 1975; Zijdenbos et al. 1994. [22] Alt & Guibas 2000. [23] Dyrland et al. 2022.

Bibliography

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

Alt, H., Guibas, L. J. (2000): *Discrete geometric shapes: matching, interpolation, and approximation*. In: Sack, Urrutia (2000): ch. 3:121–153. DOI:10.1016/B978-044482537-7/50004-8.

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

Berger, J. O. (1985): *Statistical Decision Theory and Bayesian Analysis*, 2nd ed. (Springer, New York). DOI:10.1007/978-1-4757-4286-2. First publ. 1980.

Brodersen, K. H., Ong, C. S., Stephan, K. E., Buhmann, J. M. (2010): *The balanced accuracy and its posterior distribution*. Proc. Int. Conf. Pattern Recognit. **20**, 3121–3124. DOI:10.1109/ICPR.2010.764.

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

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

Cox, R. T. (1946): *Probability, frequency, and reasonable expectation*. Am. J. Phys. **14**[1], 1–13. DOI:10.1119/1.1990764.

Dice, L. R. (1945): *Measures of the amount of ecologic association between species*. Ecology **26**[3], 297–302. DOI:10.2307/1932409.

Dyrland, K., Lundervold, A. S., Porta Mana, P. G. L. (2022): *A probability transducer and decision-theoretic augmentation for machine-learning classifiers*. Open Science Framework DOI:10.31219/osf.io/vct9y.

Fine, T. L. (1973): *Theories of Probability: An Examination of Foundations*. (Academic Press, New York). DOI:10.1016/C2013-0-10655-1.

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

Fleiss, J. L. (1975): *Measuring agreement between two judges on the presence or absence of a trait*. Biometrics **31**[3], 651–659. DOI:10.2307/2529549.

Fowlkes, E. B., Mallows, C. L. (1983): *A method for comparing two hierarchical clusterings*. J. Am. Stat. Assoc. **78**[383], 553–569. DOI:10.1080/01621459.1983.10478008.

Gilovich, T., Griffin, D., Kahneman, D., eds. (2009): *Heuristics and Biases: The Psychology of Intuitive Judgment*, 8th pr. (Cambridge University Press, Cambridge, USA). DOI:10.1017/CBO9780511808098. First publ. 2002.

Good, I. J., Toulmin, G. H. (1968): *Coding theorems and weight of evidence*. IMA J. Appl. Math. **4**[1], 94–105. DOI:10.1093/imamat/4.1.94.

Halpern, J. Y. (1999a): *A counterexample to theorems of Cox and Fine*. J. Artif. Intell. Res. **10**, 67–85. DOI:10.1613/jair.536. See also Snow (1998), Halpern (1999b).

— (1999b): *Cox's theorem revisited*. J. Artif. Intell. Res. **11**, 429–435. DOI:10.1613/jair.644. See also Snow (1998).

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

Howard, R. A. (1980): *On making life and death decisions*. In: Schwing, Albers (1980): 89–113. With discussion. DOI:10.1007/978-1-4899-0445-4_5. Repr. in Howard, Matheson (1984) pp. 481–506.

Howard, R. A., Matheson, J. E., eds. (1984): *Readings on the Principles and Applications of Decision Analysis. Vol. II: Professional Collection*. (Strategic Decisions Group, Menlo Park, USA).

— (2005): *Influence diagrams*. Decis. Anal. **2**[3], 127–143. DOI:10.1287/deca.1050.0020. First publ. 1984 in Howard, Matheson (1984) pp. 719–762.

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

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

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

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

Kahneman, D. (2011): *Thinking, Fast and Slow*. (Farrar, Straus and Giroux, New York).

Kahneman, D., Slovic, P., Tversky, A., eds. (2008): *Judgment under uncertainty: Heuristics and biases*, 24th pr. (Cambridge University Press, Cambridge). DOI:10.1017/CBO9780511809477. First publ. 1982.

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

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

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

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.

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

Metz, C. E. (1978): *Basic principles of ROC analysis*. Semin. Nucl. Med. **VIII**[4], 283–298. DOI:10.1016/S0001-2998(78)80014-2.

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

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

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.

Ramsey, F. P. (1926): *Truth and probability*. In: Ramsey (1950): ch.VII:156–198. Repr. in Kyburg, Smokler (1980), pp. 23–52. Written 1926.

Ramsey, F. P. (1950): *The Foundations of Mathematics: and other Logical Essays*. (Routledge & Kegan Paul, London). doi:10.4324/9781315887814, https://archive.org/details/in.ernet.dli.2015.46352. Ed. by R. B. Braithwaite. First publ. 1931.

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

Sack, J.-R., Urrutia, J., eds. (2000): *Handbook of Computational Geometry*. (Elsevier, Amsterdam). doi:10.1016/B978-0-444-82537-7.X5000-1.

Savage, L. J. (1972): *The Foundations of Statistics*, 2nd rev. and enl. ed. (Dover, New York). First publ. 1954.

Schwing, R. C., Albers Jr., W. A., eds. (1980): *Societal Risk Assessment: How Safe is Safe Enough?* (Springer, New York). doi:10.1007/978-1-4899-0445-4.

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.

Shannon, C. E. (1948): *A mathematical theory of communication*. Bell Syst. Tech. J. **27**[3, 4], 379–423, 623–656. https://archive.org/details/bstj27-3-379, https://archive.org/details/bstj27-4-623, http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.

Snow, P. (1998): *On the correctness and reasonableness of Cox's theorem for finite domains*. Comput. Intell. **14**[3], 452–459. doi:10.1111/0824-7935.00070.

— (2001): *The reasonableness of possibility from the perspective of Cox*. Comput. Intell. **17**[1], 178–192. doi:10.1111/0824-7935.00138.

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.

Steele, K., Stefánsson, H. O. (2020): *Decision theory*. In: *Stanford encyclopedia of philosophy*, ed. by E. N. Zalta (The Metaphysics Research Lab, Stanford). https://plato.stanford.edu/archives/win2020/entries/decision-theory. First publ. 2015.

van Rijsbergen, C. J. (1974): *Foundation of evaluation*. J. Doc. **30**[4], 365–373. doi:10.1108/eb026584.

Varoquaux, G., Cheplygina, V. (2022): *Machine learning for medical imaging: methodological failures and recommendations for the future*. npj Digit. Med. **5**[1], 48. doi:10.1038/s41746-022-00592-y.

von Neumann, J., Morgenstern, O. (1955): *Theory of Games and Economic Behavior*, 3rd ed., 6th pr. (Princeton University Press, Princeton). https://archive.org/details/in.ernet.dli.2015.215284. First publ. 1944.

Woodward, P. M. (1964): *Probability and Information Theory, with Applications to Radar*, 2nd ed. (Pergamon, Oxford). doi:10.1016/C2013-0-05390-X. First publ. 1953.

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

Zijdenbos, A. P., Dawant, B. M., Margolin, R. A., Palmer, A. C. (1994): *Morphometric analysis of white matter lesions in MR images: method and validation*. IEEE Trans. Med. Imaging **13**[4], 716–724. doi:10.1109/42.363096.

Chapter 7

A probability transducer and decision-theoretic augmentation for machine-learning classifiers

Open Science Framework DOI:10.31219/osf.io/vct9y

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. 1 June 2022; updated 1 June 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.

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 17, 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.5 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[22]. 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

[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.
[22] see Ishwaran & Zarepour 2002 on why the number of terms does not need to be infinite.

the marginal distributions for c and for y:

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

from which also the conditional distributions are easily obtained:

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

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

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 Markov-chain Monte Carlo sampling, discussed in appendix B.3. 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 $i = 1, 2, \dots$. 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}.[23] 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

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

$p(c \mid y)$ 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 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[24].

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

[24] Dyrland et al. 2022a.

is characterized by a naturally high class imbalance. The classification problem is moreover of interest to us for other ongoing research projects.

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

Two machine-learning classifiers are considered: a Random Forest and a residual Convolutional Neural Network (ResNet), details of which are given in appendix 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[25], previously used in the literature for other studies of machine-learning applications to drug discovery[26]. 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.3.

[25] Bento et al. 2014. [26] Koutsoukas et al. 2017.

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. (30). The sum contains $2^{18} \approx 260\,000$ terms.

Figure 1 shows the probabilities of classes 1 and 0 conditional on the random-forest output: $\mathrm{p}(\text{class } 1 \mid \text{output})$ and $\mathrm{p}(\text{class } 0 \mid \text{output})$. 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

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

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, $\mathrm{p}(\text{class } 1 \mid \text{outputs})$. Its extremal

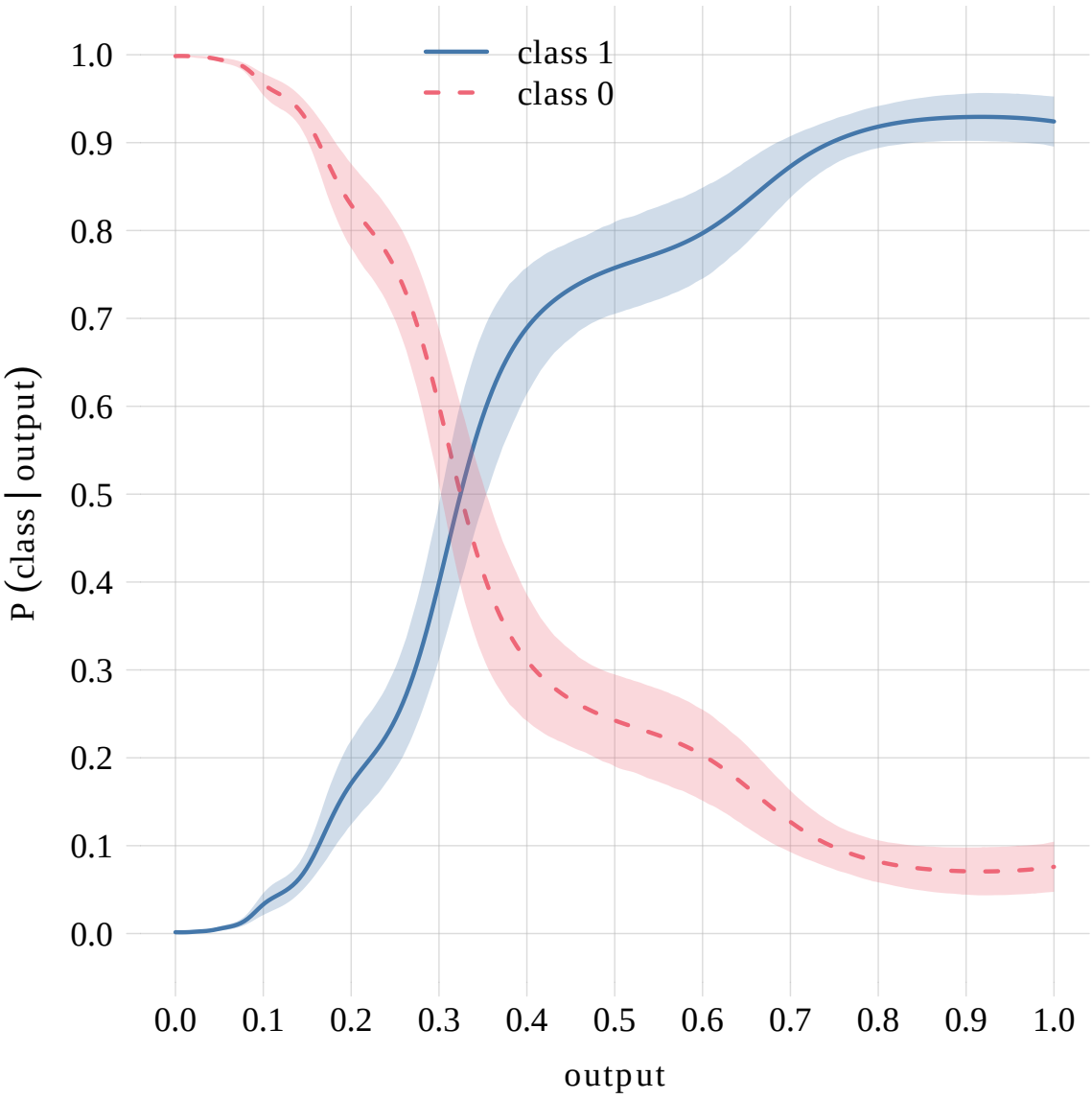


Figure 1 Probabilities of class 1 ('active', blue solid curve) and class 0 ('inactive', red dashed curve) conditional on the random-forest output. Their extremal values are 0.0014 and 0.929. The shaded region around each curve represents its 12.5%–87.5% range of possible variability if more data were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

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

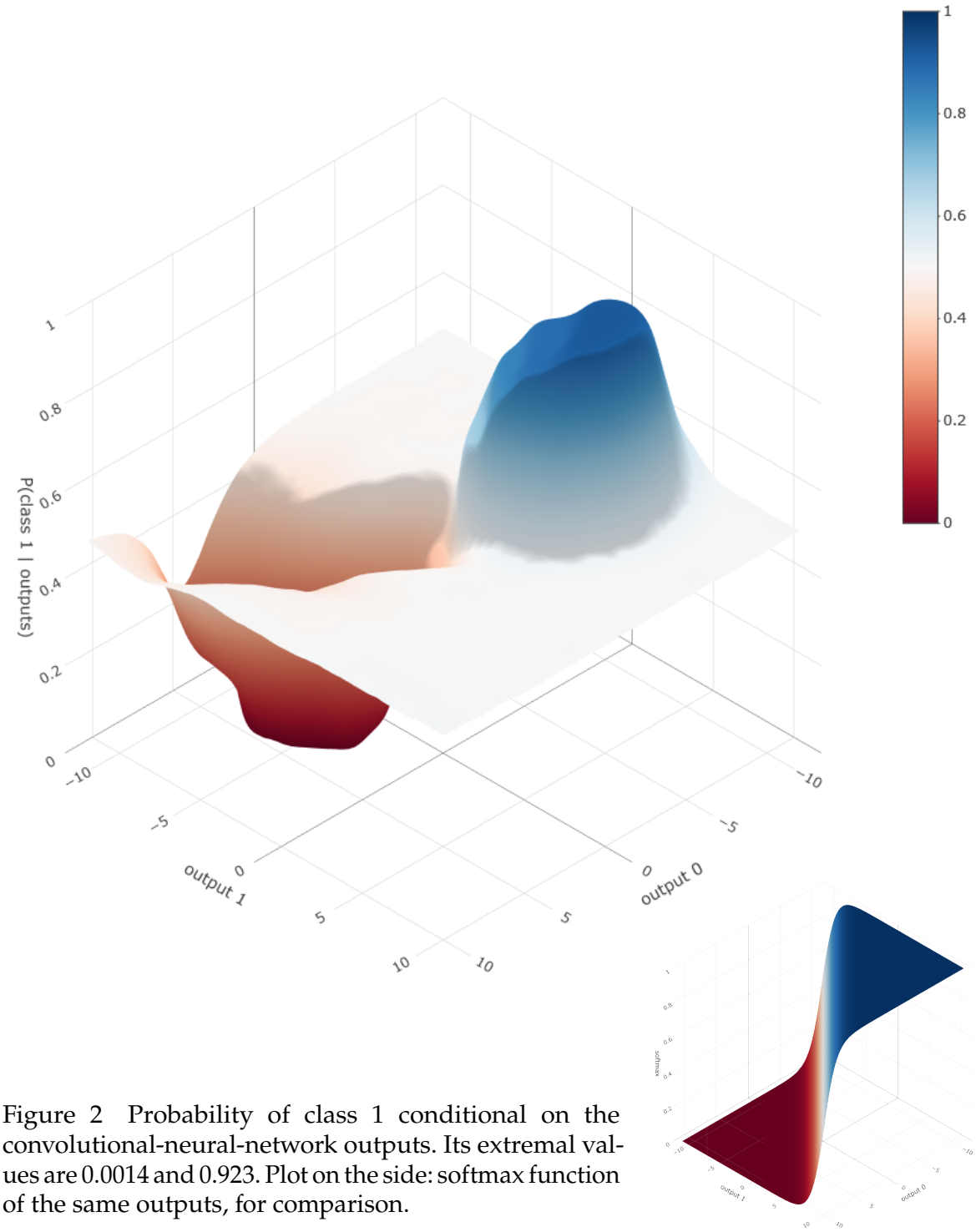


Figure 2 Probability of class 1 conditional on the convolutional-neural-network outputs. Its extremal values are 0.0014 and 0.923. Plot on the side: softmax function of the same outputs, for comparison.

neural networks[27]. 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.

[27] Gal & Ghahramani 2016.

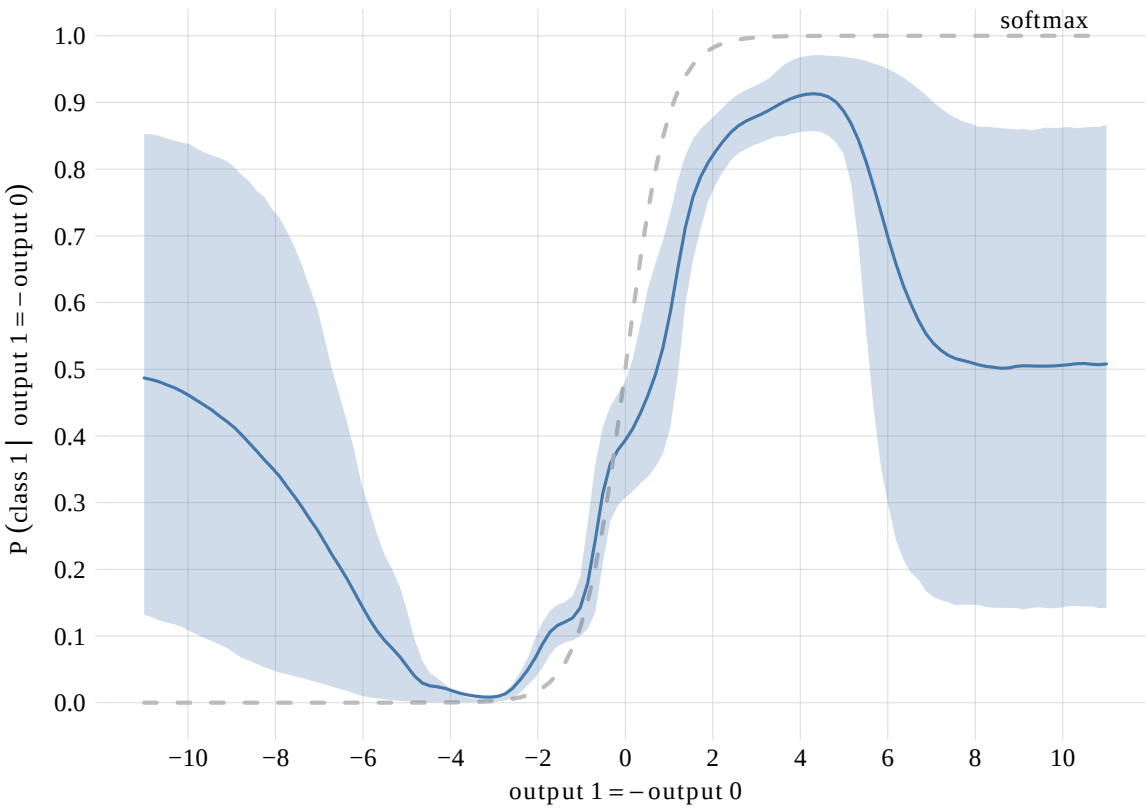


Figure 3 Cross-section of the probability surface of fig. 2 across the bisector of the II and IV quadrants of output space. The shaded region represents the 12.5%–87.5% range of possible variability upon increase of the calibration dataset. The cross-section of the softmax function (grey dashed curve)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

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 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

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

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:

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

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[28], 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

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

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, for both the random forest and the convolutional neural network, to the confusion matrix

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

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 algorithm 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 is a high cost (−10) if the true class is not 1, and on the other hand a high reward (10) if the true class is indeed 0. Now, a look at the transducer curve for the random forest, fig. 1, shows that the transducer never assigns 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[29]. It can be seen that this

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

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 our companion work[30] 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

[30] 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

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.
- 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[31].

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

[31] Dyrland et al. 2022a § 3.2.

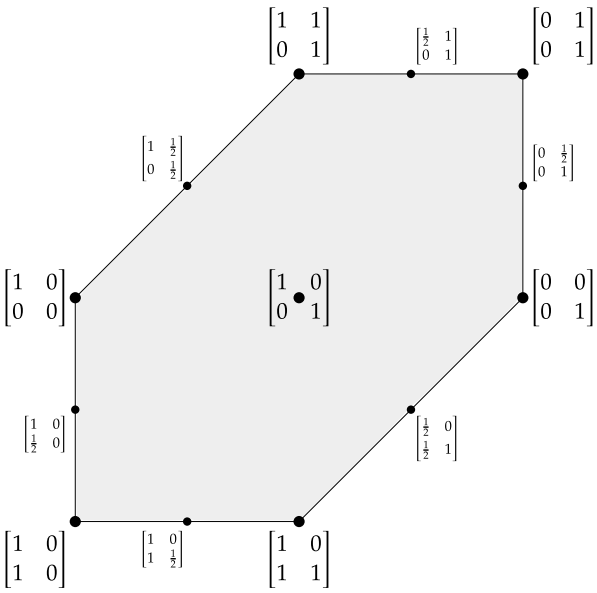


Figure 4 Space of utility matrices (modulo equivalence) for binary classification.

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.

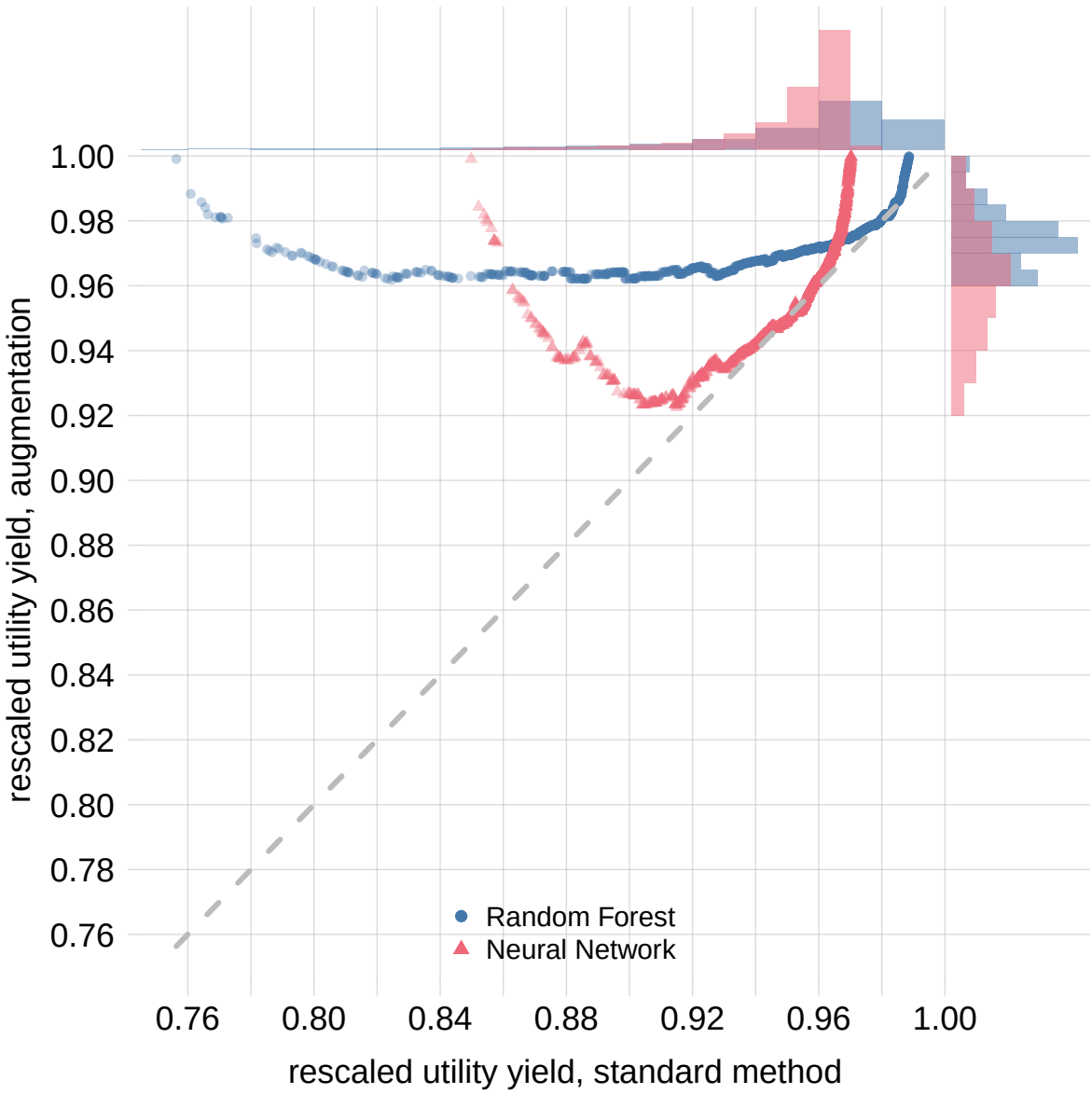


Figure 5 Rescaled utility yields obtained using the two classifiers in the standard way, vs those obtained with augmentation, for a uniform distribution of possible utility matrices over the utility-matrix space of fig. 4. The augmentation always leads to an improved utility yield, especially in cases 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).

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.4.

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.5; see especially formula (31).

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

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

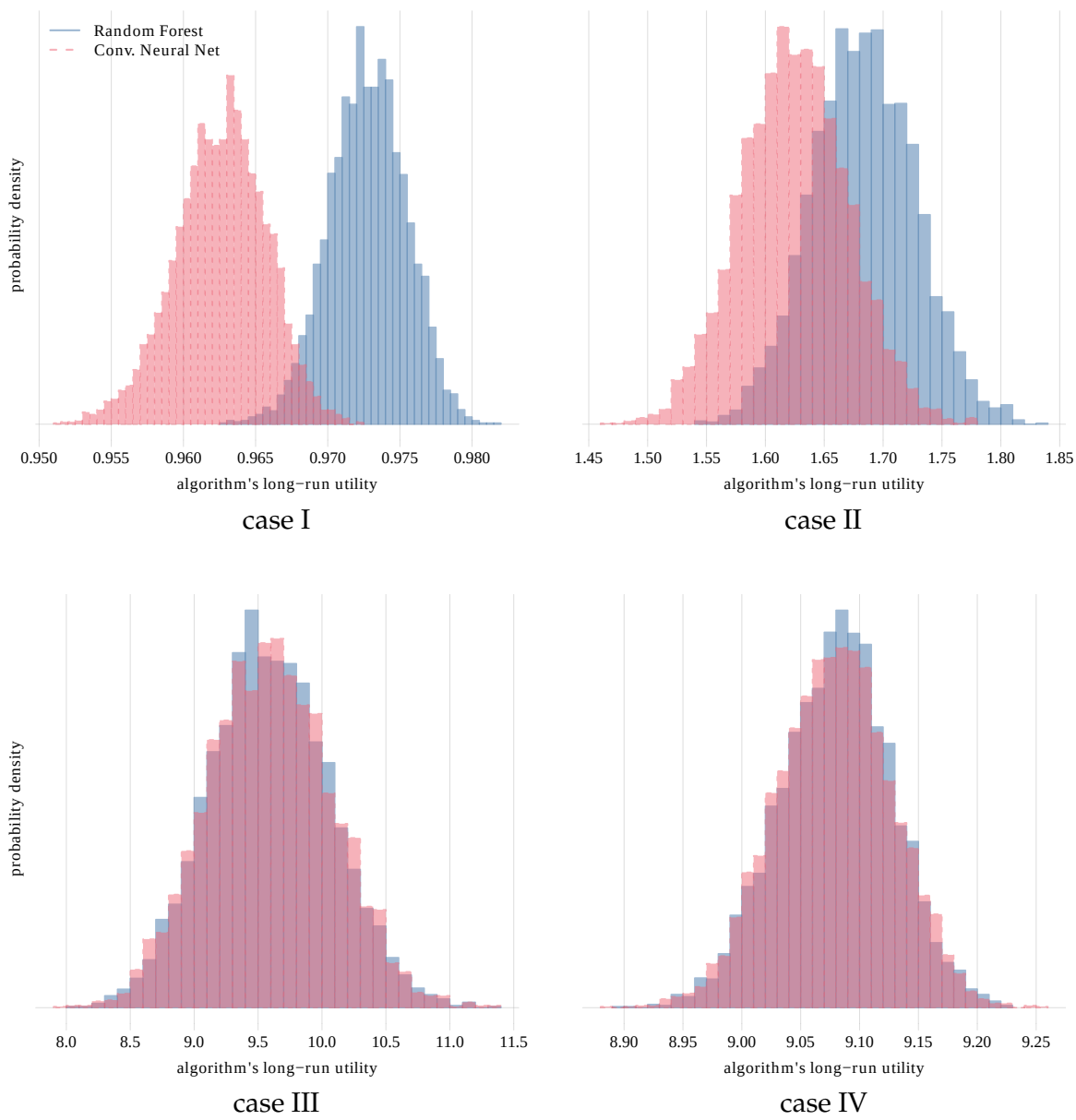


Figure 6 Probability distributions of the long-run utility yields of random forest and convolutional neural network in the four cases of § 4

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[32].

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. (32), 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[33] 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*

[32] cf. Smith & Winkler 2006. [33] Russell & Norvig 2022 § 21.2.3; Murphy 2012 § 8.6.

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

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

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[34], also called *base rates*[35], 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.

[34] Sox et al. 2013 ch. 3; Hunink et al. 2014 § 5.1. [35] Bar-Hillel 1980; Axelsson 2000.

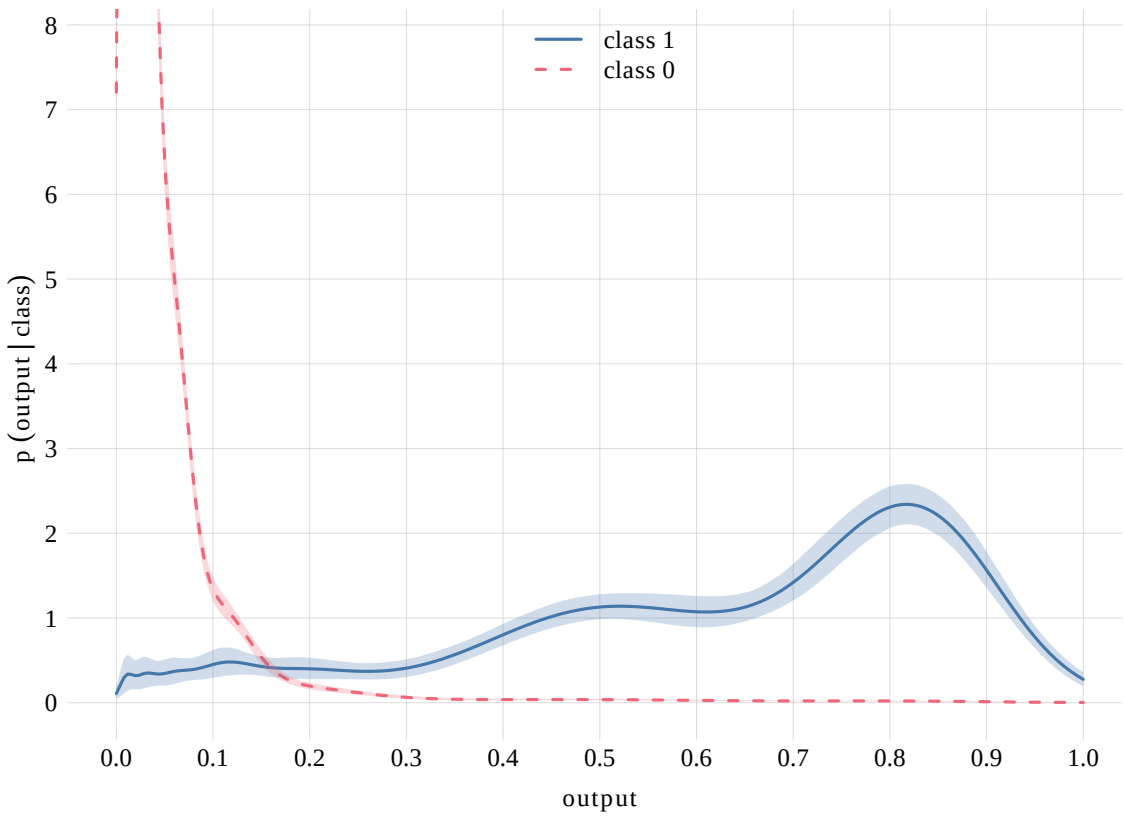


Figure 7 Probability densities of the random-forest output conditional on class 1 ('active', blue solid curve) and on class 0 ('inactive', red dashed curve, truncated). The shaded region around each curve represents its 12.5%–87.5% range of possible variability upon increase of the calibration dataset.

Figure 7 shows the 'generative' probability densities p(output | class 1), p(output | 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.

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

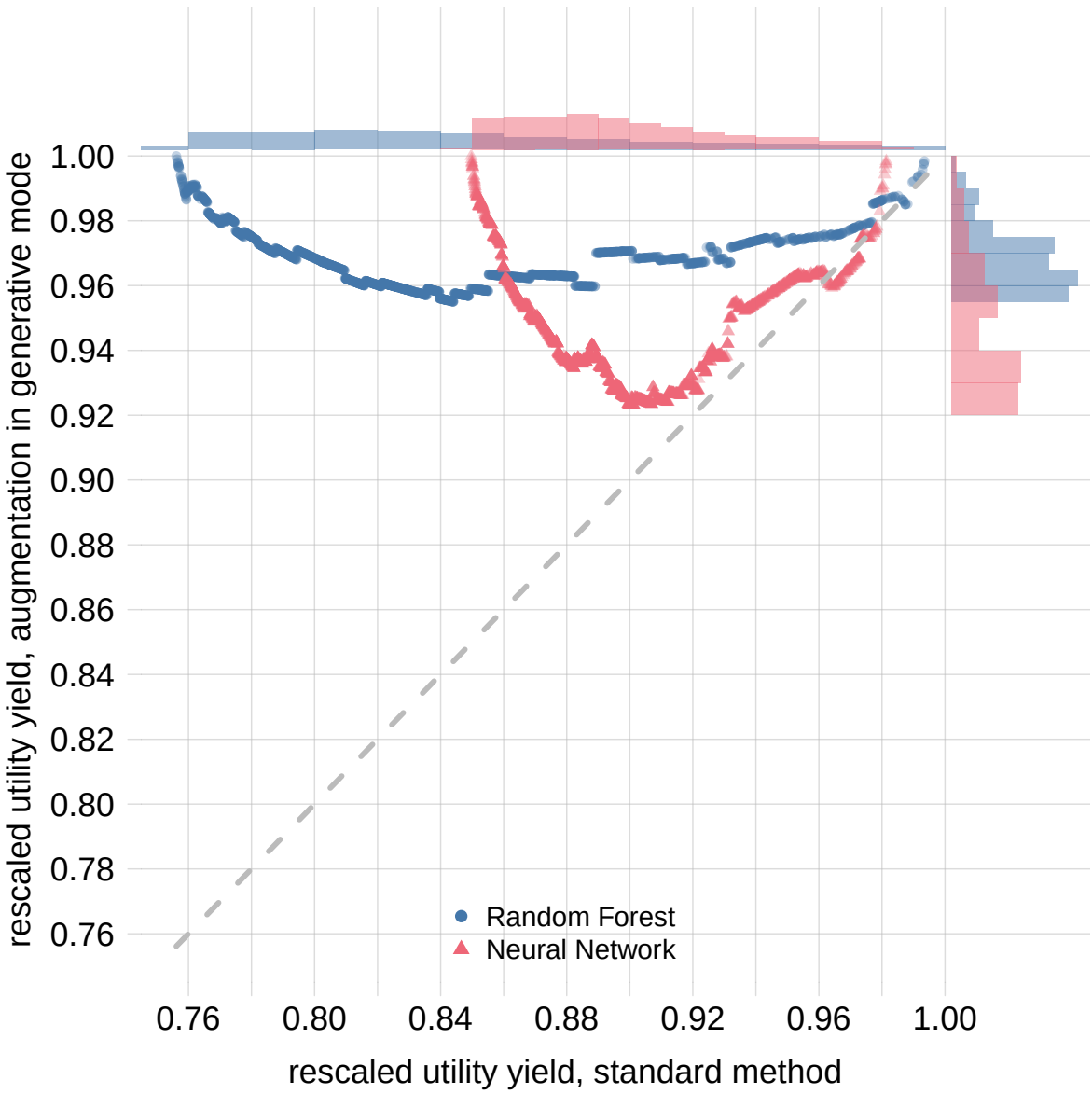


Figure 8 Rescaled utility yields obtained using the two classifiers in the standard way, vs those obtained with augmentation in generative mode, on an altered dataset with very different class balance from the training and calibration sets. The distribution of possible utility matrices is uniform over 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.

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 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[36]: Decision Theory, whose methods keep on see-sawing in machine learning[37]. 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.

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

This probabilistic quantification is not computationally expensive, can be calculated exactly by Bayesian *model-free* (non-parametric) density-regression methods[38], 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 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

[38] Dunson & Bhattacharya 2011.

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[39]. 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 factors separately.[40] 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,

[39] Elkan 2001; Correa Bahnsen et al. 2015; Ling & Sheng 2017. [40] 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.

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.

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[41]. 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)[42]. 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[43] from the dataset and converting it into a canonical graph structure by means of RdKit[44]. 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[45], 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

[41] Bento et al. 2014. [42] Rogers & Hahn 2010. [43] David et al. 2020. [44] Landrum et al. 2017. [45] 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[46]. 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[47].

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[48]. RF is an ensemble of classifying or regression trees where the majority of votes is chosen as the predicted class[49]. 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[50]. 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)[51] with 18 hidden layers trained on the well-known ImageNet dataset[52] by using the PyTorch framework[53]. ResNet has shown to outperform other pre-trained convolutional neural network models[54]. A ResNet with 34 hidden layers

[46] Green 2019. [47] Koutsoukas et al. 2017. [48] Pedregosa et al. 2011. [49] Breiman 2001. [50] Svetnik et al. 2003. [51] He et al. 2016. [52] Russakovsky et al. 2015. [53] Paszke et al. 2019. [54] 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 $d := (c, y)$.

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[55]. 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[56]. 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.

[55] Bernardo & Smith 2000 ch. 4; Dawid 2013; de Finetti 1929; 1937. [56] 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 (compare fig. 9).

B.2 Conditional probabilities

From eq. (21), which is expressed in terms of joint probabilities for c_0 and y_0, there are two ways of obtaining the probability of c_0 conditional on y_0, which we report without proof:

Exchangeable output: if the newly observed value y_0 of the output is considered to be exchangeable with (or representative of) the y values in the calibration data, then

$$\mathrm{p}(c_0 \mid y_0, D, \text{exch.}) = \frac{\mathrm{p}(c_0, y_0 \mid D)}{\mathrm{p}(y_0 \mid D)} = \frac{\int F(c_0, y_0)\, w(F \mid D)\, \mathrm{d}F}{\int F(y_0)\, w(F \mid D)\, \mathrm{d}F}\,, \tag{24}$$

where $F(y_0) = \sum_c F(c, y_0)$. This expression is effectively doing two things: first, implicitly updating the probability distribution for y, taking as new evidence the observed y_0; second, yielding the conditional distribution of c_0 given y_0.

Non-exchangeable output: if the newly observed value y_0 of the output is *not* considered to be exchangeable with the y values in the calibration data, then

$$\mathrm{p}(c_0 \mid y_0, D, \text{non-exch.}) = \int \frac{F(c_0, y_0)}{F(y_0)}\, w(F \mid D)\, \mathrm{d}F\,; \tag{25}$$

This expression does not implicitly update of the distribution for y.

The second formula should be used if new data are considered to lead to a distribution of features different from that of the calibration data – although the basic assumption that the *conditional* distributions of classes given features are the same still holds[57].

[57] see Lindley & Novick 1981 for a thorough discussion of these two cases.

The two formulae converge to one another and to the long-run conditional probability of c given y, as the number of calibration data increases. With a large number of calibration data, say hundreds or more, the values obtained from the two formulae are negligible.

B.3 Representation of the long-run distribution and Markov-chain Monte Carlo sampling

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):

$$\mathrm{p}(d_0 \mid D) = \int F(d_0)\, w(F \mid D)\, \mathrm{d}F \approx \frac{1}{T} \sum_{t=1}^{T} F_t(d_0)\,. \tag{26}$$

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

$$F(c, y) = \sum_{l} w_l\, A(c \mid \alpha_l)\, B(y \mid \beta_l)\,, \tag{27}$$

where the sum has a large number of terms[58], $\{w_l\}$ 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[59] has several advantages:

- Its marginal distributions for c and y are also of the form (27), as shown in § 2.3, and easily computable.
- Its conditional distributions for c given y and vice versa have also a form similar to eq. (27) and easily computable.
- It can be used with conjugate priors.

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

- The final probability $p(c, y)$, approximated by the sum (26), has also the form (27):

$$\mathrm{p}(c_0, y_0) \approx \sum_{t,l} \frac{w_{t,l}}{T}\, A(c_0 \mid \alpha_{t,l})\, B(y_0 \mid \beta_{t,l})\,. \tag{28}$$

  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:

$$A(c \mid \alpha) = c\ \alpha + (1 - c)\,(1 - \alpha) \equiv \begin{cases} \alpha & \text{if } c = 1, \\ 1 - \alpha & \text{if } c = 0\ ; \end{cases} \tag{29}$$

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

$$B(y \mid \beta) = \mathrm{N}(y \mid \mu, \sigma) \coloneqq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right], \tag{30}$$

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.

The samples $\{F_t(\,\cdot\,)\}$ of the sum (26) – these are samples of the distribution $w(F)$ – are obtained through Markov-chain Monte Carlo; specifically Gibbs sampling[60]. 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 $(1/{\sigma_l}^2)$.

The Markov-chain Monte Carlo sampling scheme is implemented using the R[61] package nimble[62], and uses 16 parallel chains[63]. 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[64].

[60] Neal 1993; MacKay 2005 ch. 29. [61] R Core Team 2022. [62] nimble 2021. [63] Dyrland et al. 2022b, scripts `RFmcmc.R` and `NNmcmc.R`. [64] Dyrland et al. 2022b, files `transducer_params-Random_Forest.zip` and `transducer_params-Neural_Net.zip`.

B.4 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 (26), 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. Figure 9 shows a small number of such samples for the transducer curve of the random forest of § 4 (cf. fig. 1). 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.5 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 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[65]. We here give only a sketch and refer to the works above for details.

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

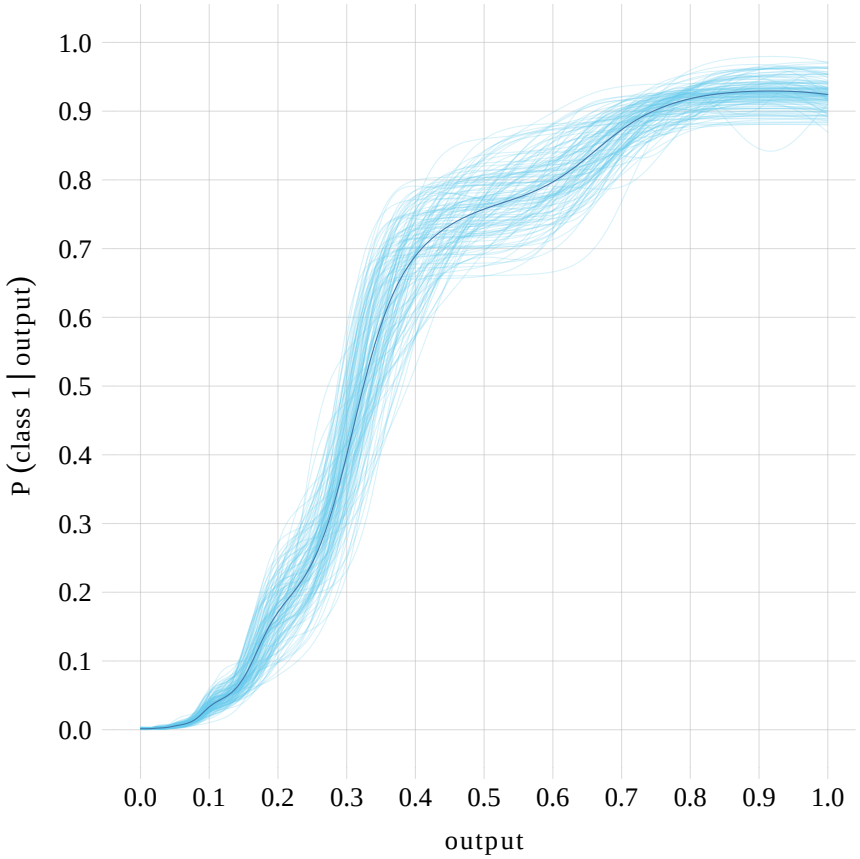


Figure 9 Samples (thin light-blue curves) of probable transducer curves (for class 1) that could be obtained if we had more calibration data in the demonstration of § 4. They are samples obtained from the distribution $w(F \mid D)$ of eq. (22). According to the probability calculus, the curve to be used for a new unit is their average (thicker dark-blue curve), which is the same as plotted in fig. 1.

Our double decision & uncertainty problem can be represented as a *decision tree*. A very simplified example is illustrated in fig. 10.

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 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

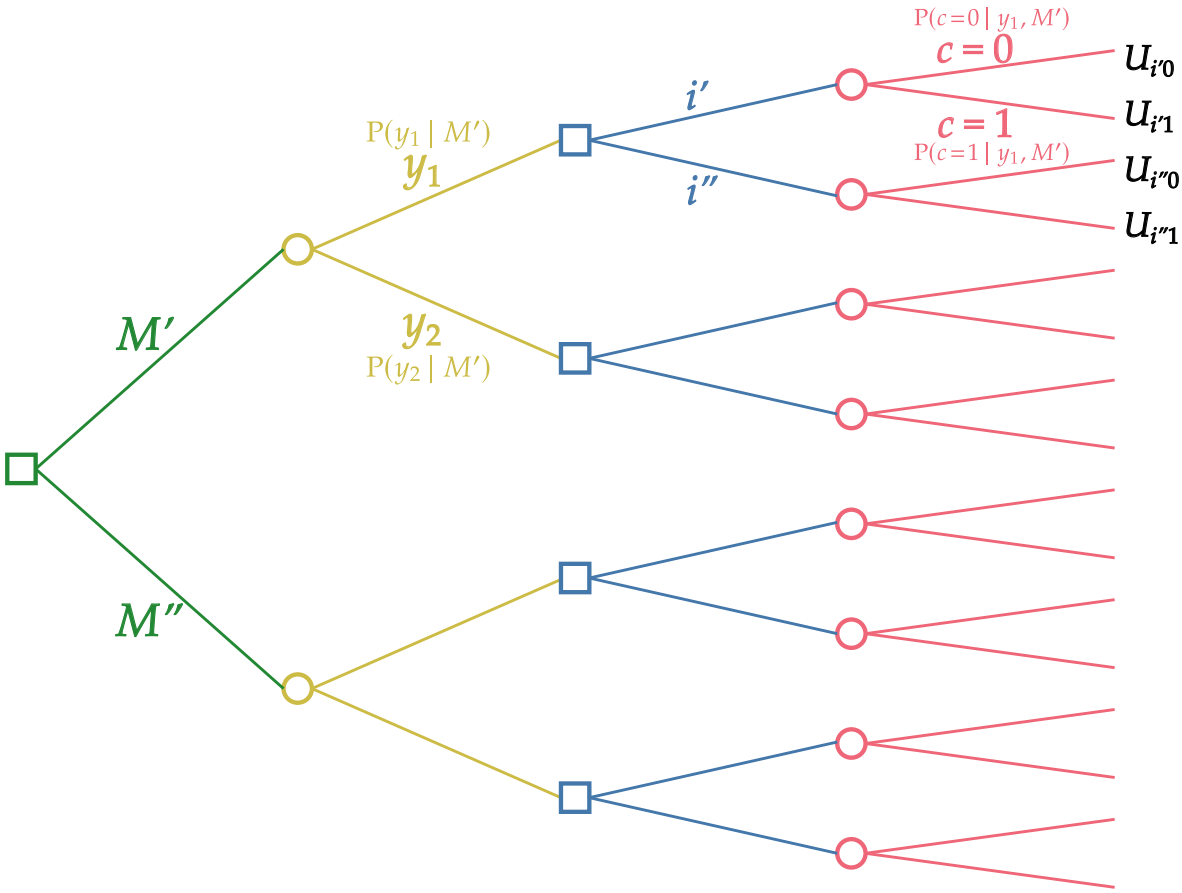


Figure 10 Decision tree for the choice of algorithm. From left to right: Decision node about the algorithm (green), uncertainty node about the algorithm's output (yellow), decision node (e.g. about class) for the inspected item (blue), uncertainty node about the class (red). Only some example nodes and branches are labelled.

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,

$$\text{utility of algorithm } M = \sum_y \left[\max_i \left\{ \sum_c U_{ic} \, \mathrm{P}(c \mid y, M) \right\} \right] \mathrm{P}(y \mid M) \,. \tag{31}$$

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.4. The long-run utility of the algorithm M is given by formula (31) 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

$$\iint \delta(u' > u'') \, \mathrm{p}(u') \, \mathrm{p}(u'') \, \mathrm{d}u' \, \mathrm{d}u'' \,. \tag{32}$$

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.

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.

— (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.