Evaluating machine learning models and their diagnostic value

Gaël Varoquaux, Olivier Colliot

Evaluating machine learning models and their diagnostic value

Gael Varoquaux[1,*] **and Olivier Colliot**[2]

[1]Soda, Inria, Saclay, France
[2]Sorbonne Université, Institut du Cerveau - Paris Brain Institute - ICM, CNRS, Inria, Inserm, AP-HP, Hôpital de la Pitié-Salpêtrière, F-75013, Paris, France
[*]Corresponding author: e-mail address: gael.varoquaux@inria.fr

Abstract

This chapter describes how to validate a machine learning model. We start by detailing the main performance metrics for different tasks (classification, regression), and how they may be interpreted, including in the face of class imbalance, varying prevalence, or asymmetric cost-benefit trade-offs. We then explain how to estimate these metrics in a unbiased manner using training, validation, and test sets. We describe cross-validation procedures –to use a larger part of the data for both training and testing– and the dangers of data leakage –optimism bias due to training data contaminating the test set. Finally, we discuss how to obtain confidence intervals of performance metrics, distinguishing two situations: internal validation or evaluation of learning algorithms, and external validation or evaluation of resulting prediction models.

Keywords: validation, performance metrics, cross-validation, data leakage, external validation

Disclaimer: this is a working paper, and is still work in progress. For any comments or missing references, please email us at gael.varoquaux@inria.fr and olivier.colliot@cnrs.fr

To appear in
O. Colliot (Ed.), *Machine Learning for Brain Disorders*, Springer

1. Introduction

A machine learning (ML) model is validated by evaluating its prediction performance. Ideally, this evaluation should be representative of how the model would perform when deployed in a real life setting. This is an ambitious goal, that goes beyond the settings of academic research. Indeed, a perfect validation would probe robustness to any possible variation of the input data which may include different acquisition devices and protocols, different practices that vary from one country to another, from one hospital to another and even from one physician to another. A less ambitious goal for validation is to provide an unbiased estimate of the model performance on new –never before seen– data similar to that used for training (but not the same data!). By similar, we mean data that has similar clinical or socio-demographic characteristics and which has been acquired using similar devices and protocols. To go beyond such *internal validity*, external validation would evaluate generalization to data from different sources (for example another dataset).

This chapter addresses the following questions. How to quantify the performance of the model? This will lead us to present, in Section 2, different performance metrics that are adequate for different ML tasks (classification, regression ...). How to estimate these performance metrics? This will lead to the presentation of different validation strategies (Section 3). We will also explain how to derive confidence intervals for the estimated performance metrics, drawing the distinction between evaluating a learning algorithm or a resulting prediction model. We will present various caveats that pertain to the use of performance metrics on medical data as well as to data leakage, which can be particularly insidious.

2. Performance metrics

Metrics allow to quantify the performance of an ML model. In this section, we describe metrics for classification and regression tasks. Other tasks (segmentation, generation, detection...) can use some of these but will often require other metrics which are specific to these tasks.

2.1 Metrics for classification

2.1.1. Binary classification

For classification tasks, the results can be summarized in a matrix called the confusion matrix (Figure 1). The confusion matrix divides the test samples into four categories, depending on their true and predicted labels:

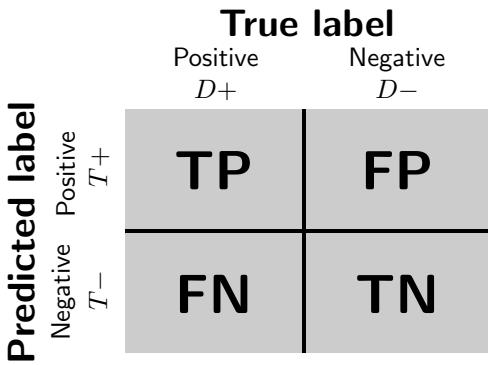


Figure 1: ***Confusion matrix*** *The confusion matrix represents the results of a classification task. In the case of binary classification (two classes), it divides the test samples into four categories, depending on their true (*eg *disease status, D) and predicted (test output, T) labels: true positives (TP), true negatives (TN), false positives (FP), false negatives (FN).*

- **True Positives (TP)**: samples for which the true and predicted labels are both 1. *Example: the patient has cancer (1) and the model classifies this sample as cancer (1)*
- **True Negatives (TN)**: samples for which the true and predicted labels are both 0. *Example: the patient does not have cancer (0) and the model classifies this sample as non-cancer (0)*
- **False Positives (FP)**: samples for which the true label is 0 and the predicted label is 1. *Example: the patient does not have cancer (0) and the model classifies this sample as cancer (1)*
- **False Negatives (FN)**: samples for which the true label is 1 and the predicted label is 0. *Example: the patient has cancer (1) and the model classifies this sample as non-cancer (0)*

Are false positives and false negatives equally problematic? This depends on the application. For instance, consider the case of detecting brain tumors. For a screening application, detected positive cases would then be subsequently reviewed by a human expert, one can thus consider that false negatives (missed brain tumor) lead to more dramatic consequences than false positives. One the opposite, if a detected tumor leads the patient to be sent to brain surgery without complementary exam, false positives are problematic and brain surgery is not a benign operation. For automatic volumetry from magnetic resonance images (MRI), one could argue that false positives and false negatives are equally problematic.

Box 1: Performance metrics for binary classification

Basic metrics

T denotes *test*: classifier output; D denotes *diseased* status.

- **Sensitivity (also called recall)**: fraction of positive samples actually retrieved.
  Sensitivity $= \frac{\text{TP}}{\text{TP+FN}}$ Estimates $P(T+ \mid D+)$
- **Specificity**: fraction of negative samples actually classified as negative.
  Specificity $= \frac{\text{TN}}{\text{TN+FP}}$ Estimates $P(T- \mid D-)$
- **Positive predictive value (PPV, also called precision)**: fraction of the positively classified samples which are indeed positive.
  PPV $= \frac{\text{TP}}{\text{TP+FP}}$ Estimates $P(D+ \mid T+)$
- **Negative predictive value (NPV)**: fraction of the negatively classified samples which are indeed negative.
  NPV $= \frac{\text{TN}}{\text{TN+FN}}$ Estimates $P(D- \mid T-)$

Summary metrics

- **Accuracy**: fraction of the samples correctly classified.
  Accuracy $= \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$
- **Balanced accuracy (BA)**: accuracy metric that accounts for unbalanced samples.
  BA $= \frac{\text{Sensitivity+Specificity}}{2}$
- **F_1-score**: harmonic mean of PPV (precision) and sensitivity (recall).
  $F_1 = \frac{2}{\frac{1}{\text{PPV}}+\frac{1}{\text{Sensitivity}}} = \frac{2\text{TP}}{2\text{TP+FP+FN}}$
- **Matthews correlation coefficient (MCC)**. MCC=1 for perfect classification, MCC=0 for random classification, MCC=-1 for perfectly wrong classification.
  MCC $= \frac{\text{TP}\times\text{TN}-\text{FP}\times\text{FN}}{\sqrt{(\text{TP+FP})\times(\text{TP+FN})\times(\text{TN+FP})\times(\text{TN+FN})}}$
- **Markedness** $= \frac{\text{TP}}{\text{TP+FP}} - \frac{\text{FP}}{\text{FP+TN}} = \text{PPV} + \text{NPV} - 1$
- **Area under the receiver operating characteristic curve (ROC AUC).**
- **Area under the precision-recall curve (PR AUC).**

Multiple performance metrics can be derived from the confusion matrix, all easily computed using `sklearn.metrics` from scikit-learn [1]. They are summarized in Box 1. One can distinguish between *basic metrics* which only focus on false positives or false negatives and *summary metrics* which aim at providing an overview of the performance with a single metric.

The performance of a classifier is characterized by pairs of basic metrics: either sensitivity and specificity, or PPV and NPV, which characterize respectively the probability of the test given the diseased status or vice versa (see Box 1). Note that each basic metric characterizes only the behavior of the classifier on the positive class ($D+$) or the negative class ($D-$); thus measuring both sensitivity *and* specificity, or PPV *and* NPV is important. Indeed a classifier always reporting a positive prediction would have a perfect sensitivity, but a disastrous specificity.

Simple summaries and their pitfalls. It is convenient to use summary metrics which provide a more global assessment of the performance, for instance to select a "best" model. However, as we will see, summary metrics, when used in isolation, can be lead to erroneous conclusions. The most widely used summary metric is arguably accuracy. Its main advantage is a natural interpretation: the proportion of correctly classified samples. However, it is misleading when the data is imbalanced. Let us for instance consider a dataset with 10 cancer samples and 990 non-cancer samples. A trivial majority classifier which decides that cancer does not exist achieves 99% accuracy. Balanced accuracy helps for imbalanced samples. However, balanced accuracy also comes with its loopholes. Indeed a high balanced accuracy does not always mean that individuals classified as diseased are likely to be so. Let us consider a diagnostic test for a disease which has a sensitivity of 99% and a specificity of 90% (and thus a balanced accuracy of 94.5%). Suppose that a given person takes the test and that the test is positive. At this point, we do not have enough information to compute the probability that the person actually has the disease.

The probability that the person has the disease is given by the PPV, related to the sensitivity and the specificity by Bayes' rule:

$$P(\overbrace{D+}^{\text{Diseased}} \mid \underbrace{T+}_{\text{Test positive}}) = \frac{\text{sensitivity} \times \text{prevalence}}{(1 - \text{specificity}) \times (1 - \text{prevalence}) + \text{sensitivity} \times \text{prevalence}}.$$

Bayes' rule thus shows that we must account for the prevalence: the proportion of the people with the disease in the target population, the population in which the test is intended to be applied. The target population can be the general population for a screening test. It could be the

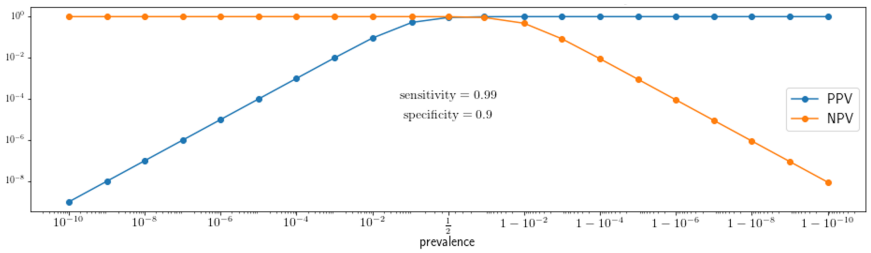


Figure 2: *NPV and PPV as functions of prevalence when the sensitivity and the specificity are fixed (image courtesy of Johann Faouzi).*

population of people with memory complaints for a test aiming to diagnose Alzheimer's disease. Now, suppose that the prevalence is low, which will often be the case for a screening test in the general population. For instance, prevalence = 0.001. This leads to $P(D+ \mid T+) = 0.0098 \approx 1\%$. So, if the test is positive, there is only 1% chance that the patient has the disease. Even though our classifier has seemingly good sensitivity, specificity, and balanced accuracy, it is not very informative on the general population. The PPV and NPV readily give the information of interest: $P(D+ \mid T+)$ and $P(D- \mid T-)$. However, they are not natural metrics to report a classifier's performance because, unlike sensitivity and specificity, they are not intrinsic to the test (in other words the trained ML model) but also depend on the prevalence and thus on the target population (Figure 2).

Summary metrics for low prevalence. The F_1 score is another summary metric, built as the harmonic mean of the sensitivity (recall) and PPV (precision). It is popular in machine learning but, as we will see, it also has substantial drawbacks. Note that it is equal to the Dice coefficient used for segmentation. Given that it builds on the PPV rather than the specificity to characterize retrieval, it accounts slightly better for prevalence. In our example, the F_1 score would have been low. The F_1 score can nevertheless be misleading if the prevalence is high. In such a case, one can have high values for sensitivity, specificity, PPV, F_1 score but a low NPV. A solution can be to exchange the two classes. The F_1 score becomes informative again. Those shortcomings are fundamental, as the F_1 score is completely blind to the number of true negative, TN. This is probably one of the reasons why it is a popular metric for segmentation (usually called Dice rather than F_1) as in this task TN is almost meaningless (TN can be made arbitrarily large by just changing the field of view of the image). In addition this metric has no simple link to the probabilities of interest, even more so after switching classes.

Another option is to use Matthews Correlation Coefficient (MCC).

The MCC makes full use of the confusion matrix and can remain informative even when prevalence is very low or very high. However, its interpretation may be less intuitive than that of the other metrics. Finally, *markedness* [2] is a seldom known summary metric that deals well with low-prevalence situations as it is built from the PPV and NPV (Box 1). Its drawback is that it is as much related to the population under study as to the classifier.

As we have seen, it is important to distinguish metrics which are intrinsic characteristics of the classifier (sensitivity, specificity, balanced accuracy) from those which are dependent on the target population and in particular of its prevalence (PPV, NPV, MCC, markedness). The former are independent of the situation in which the model is going to be used. The latter informs on the probability of the condition (the output label) given the output of the classifier; but they depend on the operational situation, and in particular on the prevalence. The prevalence can be variable (for instance the prevalence of an infectious disease will be variable across time, the prevalence of a neurodegenerative disease will depend on the age of the target population) and a given classifier may be intended to be applied in various situations. This is why the intrinsic characteristics (sensitivity and specificity) need to be judged according to the different intended uses of the classifier (e.g. a specificity of 90% may be considered excellent for some applications while it would be considered unacceptable if the intended use is in a low prevalence situation).

Metrics for shifts in prevalence. Odds enable designing metrics that characterize the classifier but are adapted to target populations with a low prevalence. Odds are defined as the ratio between the probability that an event occurs to the probability this event does not occur: $O(a) = \frac{P(a)}{1-P(a)}$. Ratios between odds can be invariant to the sampling frequency (or prevalence) of a –see appendix A.1 for an introduction to odds and their important properties. For this reason, they are often used in epidemiology. A classifier can be characterized by the ratio between the pre-test and post-test odds, often called the *positive likelihood ratio*: $\text{LR+} = \frac{O(D+|T+)}{O(D+)} = \frac{\text{sensitivity}}{1-\text{specificity}}$. This quantity depends only on sensitivity and specificity, properties of the classifier only, and not of the prevalence on the study population. Yet, given a target population, post-test odds can easily be obtained by multiplying LR+ by pre-test odds, itself given by prevalence: $O(D+) = \frac{\text{prevalence}}{1-\text{prevalence}}$. The larger the LR+, the more useful the classifier and a classifier with LR+ = 1 or less brings no additional information on the likelihood of the disease. An equivalent to LR+ characterizes the negative class: controling on "T-" instead of "T+" gives the *negative likelihood ratio*: $\text{LR-} = \frac{1-\text{sensitivity}}{\text{specificity}}$; and low values of LR- (below

1) denote more useful predictions. These metrics, LR+ and LR- are very useful in a situation common in biomedical settings where the only data available to learn and evaluate a classifier is a *study* population with nearly balanced classes, such as a case-control study, while the target application –the general population– is one with a different prevalence (e.g. a very low prevalence) or when the intended use considers variable prevalences.

Multi-threshold metrics. Many classification algorithms output a continuous value which is then thresholded to get a binary label. When the output is a probability, one often simply uses a threshold of 0.5. However, there are cases where one is interested to study the performance for varying thresholds on the output. The two main tools for that purpose are the receiver operating characteristic (ROC) curve and the precision-recall (PR) curve. The ROC curve plots the Sensitivity as a function of 1 − Specificity (Figure 3). It can be again summarized with a single value: the area under the ROC curve (ROC AUC). The ROC AUC has a probabilistic interpretation: it is the probability that a positive sample has a higher classification score (as positive) than a negative sample. A perfect classification corresponds to a ROC AUC of 1 and a random classification to a ROC AUC of 0.5. While chance remains 0.5 whatever the class imbalance, the ROC curve becomes less interesting for highly imbalanced classes, because a seamingly small difference on specificity or sensitivity may make a large difference to the application, but not change much the ROC curve. For this reason, it is often complemented with the precision-recall (PR) curve which focuses on the minority class. The PR curve plots the Precision (also called PPV) as a function of Recall (also called Sensitivity) (Figure 4). It can also be summarized using a single

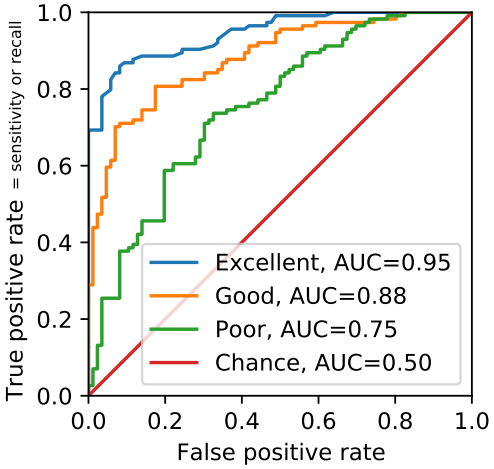


Figure 3: ***ROC curve*** *for different classifiers. AUC denotes the Area Under the Curve, typically used to extract a number summarizing the ROC curve.*

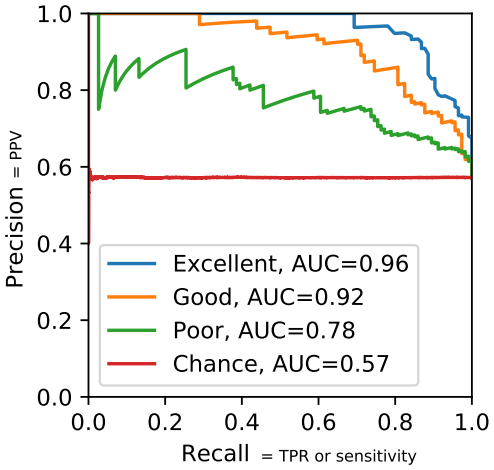


Figure 4: ***Precision-Recall curve*** *for different classifiers. AUC denotes the Area Under the Curve, often called* average precision *here. Note that the chance level depends on the class imbalance (or prevalence), here 0.57.*

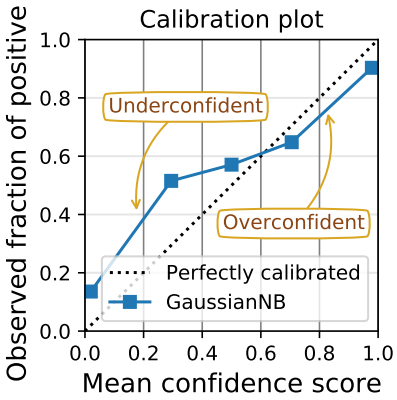
measure: the PR AUC. As for the ROC AUC, a perfect classification corresponds to a value of 1. However, unlike for ROC AUC, a dummy classification does not necessary leads to a value of 0.5. It depends on the prevalence.

Confidence scores and calibration. It can be useful to interpret a non-thresholded classifier score as a confidence score or a probability, for instance to balance cost and benefits when the prediction is used to decide on an intervention [4]. But a continuous score by itself does not warrant such interpretation: a classifier may be over-confident, under-confident, or have uneven scores over the population, even for good binary decisions. Two types of metrics, detailed in Box 2, are useful to evaluate continuous outputs as probabilities: the *Expected calibration error* (ECE) and the *Brier score*. The ECE measures whether, on samples predicted with a score s, the error rate is indeed s, in which case the classifier is said to be calibrated. The *Brier score* is minimal when the classifier score is the true probability of the class given the data for an individual, for instance the probability of presence of a tumor given the image. These two notions are similar, but controlling true probabilities, as with a Brier score, is more stringent and more useful to the practitioner. The ECE qualifies probabilities that are mostly independent of prediction performance, while the Brier score captures the ability to give good probabilistic prediction of the output, accounting both for the quality of probabilities and corresponding binary decisions. For any classification problem, there exists many classifiers with 0 expected calibration errors, including some with very poor predictions. On the other hand, even the best possible prediction has a non-zero Brier score, unless the output is a deterministic function of the data. The Brier skill score, a variant of the Brier

Box 2: Assessing confidence scores and calibration

Expected calibration error (ECE): *average classifier error*

It is computed by considering K bins of confidence scores and comparing the observed fraction of positives to the mean confidence score. The ECE itself is then the average over the bins: $ECE = \sum_{i=1}^{K} P(i) \cdot |f_i - s_i|$ where f_i is the observed fraction of positive instances in bin i, s_i is the mean of classifier scores for the instances in bin i, and P(i) is the fraction of all instances that fall into bin i [3].



Example for a Gaussian Naive Bayes classifier (GaussianNB).

Metrics on individual probabilities: *error on* $P(y|X)$

$$\textbf{Brier score} = \sum_i (\hat{s}_i - y_i)^2$$

Observed (binary) label; Confidence score

Minimal for $\hat{s} = P(y|X)$

$$\textbf{Brier skill score} = 1 - \frac{\text{Brier}(\hat{s}, y)}{\text{Brier}(\bar{y}, y)}$$

Class prevalence

A value of 1 means a perfect prediction while a value of 0 means that the confidence scores are not more informative than the class prevalence.

score is often used to assess how far a predictor is from the best possible prediction, more independent of the intrinsic uncertainty in the data. The Brier skill score is a rescaled version of the Brier score taking as a reference a reasonable baseline: 1 is a perfect prediction, while negative values mean predictions worse than guessing from class prevalence.

To conclude When assessing a classifier:

- Always look at all the individual metrics: false positives and false negatives are seldom equivalent. Understand the medical problem to known the right tradeoff [4]
- Never trust a single summary metric (accuracy, balanced accuracy, ROC AUC . . .).

- Consider the prevalence in your target population. It may be that the prevalence in your testing sample is not representative of that of the target population. In that case, aside from LR+ and LR-, performance metrics computed from the testing sample will not be representative of those in the target population.

2.1.2. Multi-class classification

When there are multiple classes to distinguish, the main difference with two-class classification is that the problem can no longer be separated into a positive class (typically individuals with the medical condition of interest) and a negative class (individuals without). As a consequence, sensitivity and specificity no longer have a meaning for the whole data, nor do F_1-score, or the ROC or precision recall curves. Accuracy is still defined and easy to compute, but still suffers from its common drawbacks, in particular that it may not be straightforward to interpret in the face of class imbalance.

A classic approach is to aggregate metrics for binary settings considering successively each class as the positive instances and all the others as the negatives, in a form of "one versus all". There are different approaches to averaging the results for each class. *Macro* averaging computes the metric, for instance the ROC AUC, for each class, and then averages the results. One drawback is that it may put too much emphasis on classes that are more infrequent. *Weighted* or *micro* averaging combine the results of the different classes weighing by the number of instance of each class. The difference between the two is that *weighted* averaging computes the average of the metric weighted by the number of true instances for each class, while *micro* averaging computes the metric by adding the number of TP (resp. TN, FP, FN) across all classes.

Inspecting the confusion matrix extended to multi-class settings gives a interesting tool to understand errors: it displays how many times a given true class is predicted as another (Figure 5). A perfect prediction has non-zero entries only on the diagonal. The confusion matrix may be interesting to reveal which classes are commonly confused, as its name suggests. In our example, instances that are actually of class C2 are often predicted as of class C3.

Multilabel classification. Multilabel settings are when the multiple classes are not mutually exclusive: for instance if an individual can have multiple pathologies. The problem is then to detect the presence or absence of each label for an individual. In terms of evaluation, multilabel settings can be understood as several binary-classification problems, and thus the corresponding metrics can be used on each label. As in the

Figure 5: ***Multi-class confusion matrix****, for a 3-class problem, C1, C2, C3. Each entry gives the number of instances predicted of a given class, knowing the actual class. A perfect prediction would give non-zero entries only on the diagonal.*

		Predicted		
		C1	C2	C3
True	C1	133	0	0
	C2	0	107	36
	C3	0	0	92

multi-class settings, there are different ways to average the results for each label –macro, micro...–, which put more or less emphasis on the rare labels.

2.2 Metrics for regression

In regression settings, the outcome to predict y is continuous, for instance an individual's age, cognitive scores, or glucose level. Corresponding error metrics gauge how far the prediction $\hat{y}$ is from the observed y.

$R2$ score The go-to metric here is typically the $R2$ score, sometimes called explained variance –however, the term $R2$ score should be preferred, as some authors define explained variance as ignoring bias. Mathematically, the $R2$ score is the fraction of variance of the outcome y explained by the prediction $\hat{y}$, relative to the variance explained by the mean $\overline{y}$ on the test set:

$$R2 = 1 - \frac{\mathrm{SS}(y - \hat{y})}{\mathrm{SS}(y - \overline{y})}$$

where SS is the sum of squares on the test data. A strong benefit of this metric is that it comes with a natural scale: an $R2$ of 1 implies perfect prediction, while an $R2$ of zero implies a trivial and not very useful prediction. Note that chance-level predictions (as obtained for instance by learning on permuted y) yield slightly negative predictions: indeed even when the data does not support a prediction of y –as in chance settings–, it is impossible to estimate the mean y perfectly and predictions will be worse than the actual mean. In this respect, the $R2$ score has a different behavior in machine learning settings compared to inferential statistics settings not focused on prediction: *in sample* (for inferential statistics) versus *out of sample* settings (for machine learning). Indeed, when the mean of y is computed on the same data as the model, the $R2$ score is positive and is the square of the correlation between y and $\hat{y}$. This is not the case in predictive settings, and the correlation between y and $\hat{y}$ should not be used to judge the quality of a prediction

[5], because it discards errors on the mean and the scale of the prediction, which are important in practice.

Absolute error measures. Reporting only the $R2$ score is not sufficient to characterize well a predictive model. Indeed, the $R2$ score depends on the variance of the outcome y in the study population, and thus does not enable comparing predictive models on different samples. For this purpose, it is important to report also an absolute error measure. The root mean square error (RMSE) and the mean absolute error (MAE) are two of such measures that give an error in the scale of the outcome: if the outcome y is an age in years, the error is also in years. The mean absolute error is easier to interpret. Compared to the root mean square error, the mean absolute error will put much less weight on some rare large deviations. For instance, consider the following prediction error (on 11 observations):

$$error = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 100]$$
$$\text{MAE} = 10 \qquad \text{RMSE} \approx 30.17$$

Note that if the error was uniformly equal to the same value (10, for instance), both measures would give the same result.

Assessing the distribution of errors. The difference between the mean absolute error and the root mean square error arises from the fact that both measures account differently for the tails of the distribution of errors. It is often useful to visualize these errors, to understand how they are structured. Figure 6 shows such visualization: predicted y as a function of observed y. It reveals that for large values of y, the predictive model has a larger prediction error, but also that it tends to undershoot: predict a value that underestimates the observed value. This aspect of the prediction error is not well captured by the summary metrics because there are comparatively much less observations with large y.

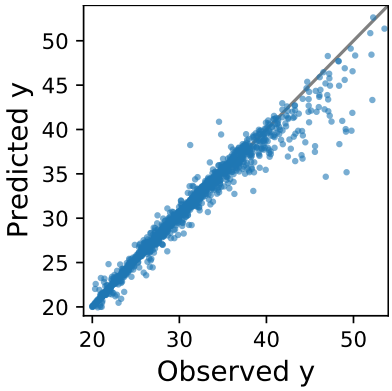


Figure 6: ***Visualizing prediction errors*** *– plotting the predicted outcome as a function of the observed one enables to detect structure in the error beyond summary metric. Here the error increases for large values of y, for which there is also a systematic undershoot.*

Concluding remarks on performance metrics. Whether it is in regression or in classification, a single metric is not enough to capture all aspects of prediction performance that are important for applications. Heterogeneity of the error, as we have just seen in our last example, can be present not only as a function of prediction target, but of any aspect of the problem, for instance the sex of the individuals. Problems related to *fairness*, where some groups (e.g. demographic, geographic, socio-economic groups) suffer more errors than others, can lead to loss of trust or amplification of inequalities [6]. For these reasons it may be important to also report error metrics on relevant subgroups, following common medical-research practice of stratification.

3. Evaluation strategies

The previous section detailed metrics for assessing the performance of a ML model. We now focus on how to estimate the expected prediction performance of the model with these metrics. Importantly, we draw the difference between evaluating a learning procedure, or learner, and a learned model. While these two questions are often conflated in the literature, the first one must account for uncontrolled fluctuations in the learning procedure, while the second one controls a given model on a target external population. The first question is typically of interest to the methods researcher, to conclude on learning procedures, while the second is central to the medical research, to conclude on the clinical application of a model.

3.1 Evaluating a learning procedure

We first focus on assessing the expected performance of a learning procedure on data drawn from a given population. Here, the model is validated on data with similar characteristics to the one used for training, a validation sometimes called *internal validation*. Most importantly *performance should not be evaluated using the same data that was used for training* [5]. Therefore, the first step is to split the data into a training set and a testing set. This should be done before starting any work on the data, be it training a ML model or even doing simple statistics for identifying interesting features. Splitting the data can be done using `sklearn.model_selection.train_test_split` or `sklearn.model_selection.ShuffleSplit(n_splits=1)` from scikit-learn. When one simply performs a single split of the data into training and testing set, the validation method is called "hold-out". One should nevertheless check that the training and testing

sets have similar characteristics. More precisely, we want the output variable distribution to be approximately the same in the training and testing sets. This is called stratification. For instance, for classification, the proportion of diseased individuals should approximately be the same in the two sets. To that purpose, use `StratifiedShuffleSplit(n_splits=1)`. In medical applications, it is recommended to control not only for the disease status but also for other variables, such as sociodemographic information (age, sex, ...) or some relevant clinical variables. It will often be difficult (and it is not even necessary) to obtain almost identical distributions between training and testing sets. In practice, it is often sufficient to have similar means and variances for continuous variables and similar proportions for categorical variables. The first two rows of Figure 7 illustrate the concepts of "hold-out" and stratification.

3.1.1. Cross-validation

The split between train and test set is arbitrary. With the same machine-learning algorithm, two different data splits will lead to two different observed performances, both of which are noisy estimates of the expected generalization performance of prediction models built with this learning procedure. A common strategy to obtain better estimates consists in performing multiple splits of the whole dataset into training and testing set: a so called *cross-validation* loop. For each split, a model is trained using the training set and the performances are computed using the testing set. The performances over all the testing sets are then aggregated. Figure 7 displays different cross-validation methods. k-fold cross-validation consists in splitting the data into k sets (called folds) of approximately equal size. It ensures that each sample in the data set is used exactly once for testing. Stratified k-fold cross-validation can be performed using `sklearn.model_selection.StratifiedKFold`.

In each split, ideally, one would want to have a large training set, because it usually allows training better performing models, and a large testing set, because it allows a more accurate estimation of the performance. But the dataset size is not infinite. Splitting out 10 to 20% for the test set is a good trade off [7]; which amounts to k=5 or 10 in a k-fold. With small datasets, to maximize the amount of train data, it may be tempting to leave out only one observation, in a so-called *leave-one-out* cross-validation. However, such depletion of the test set gives overall worse estimates of the generalization performance. Increasing the number of splits is however useful, thus another strategy consists in performing a large number of random splits of the data, breaking from the regularity of the k-fold. If the number of splits is suf-

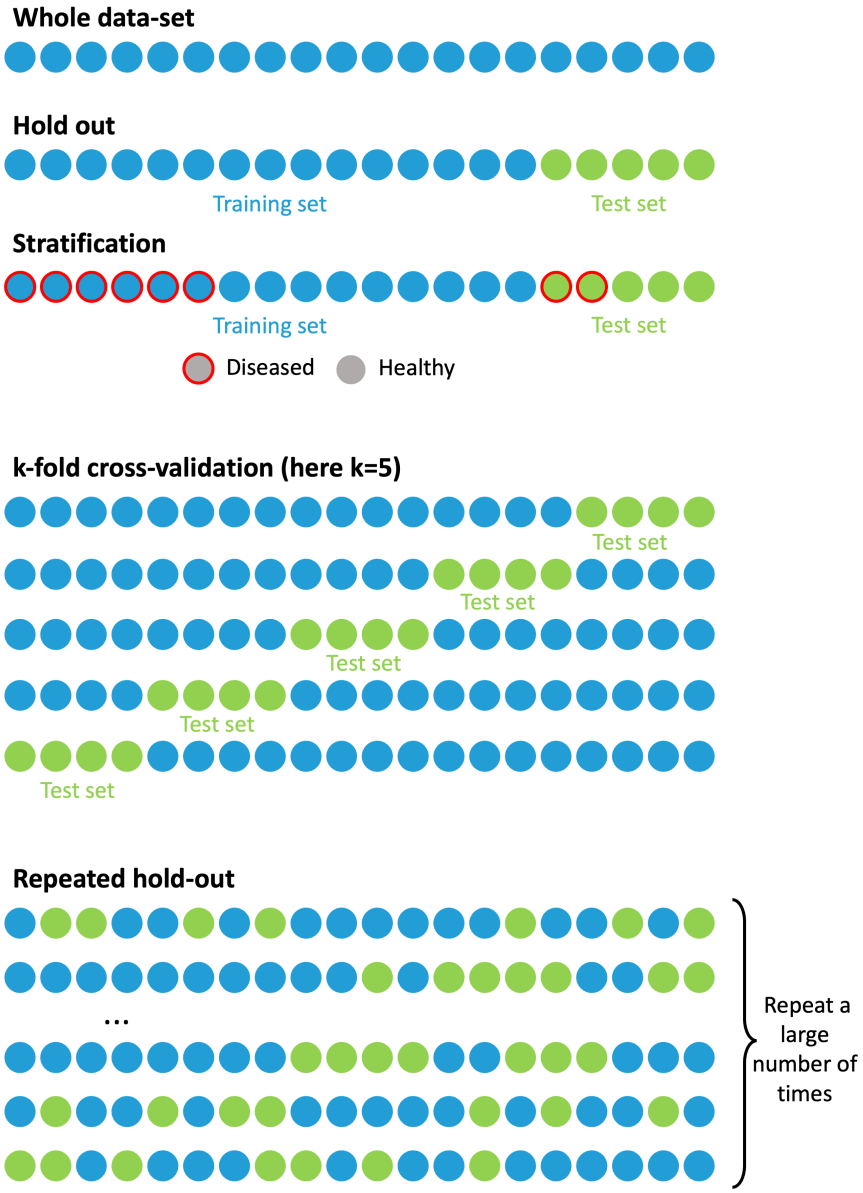


Figure 7: *Different validation methods, from top to bottom. The first method, called "hold-out", involves a single split of the dataset into training and testing sets. It is thus not a cross-validation method. Stratification is the procedure which controls that the output variable (for instance disease vs healthy) has approximately the same distribution in the training and testing set. k-fold cross-validation consists in splitting the data into k sets (called folds) of approximately equal size. Repeated hold-out consists in performing a large number of random splits of the data.*

ficiently large, all samples will be approximately used the same number of times for training and testing. This strategy can be done using `sklearn.model_selection.StratifiedSuffleSplit(n_splits)` and is called "Repeated hold-out" or "Monte-Carlo cross-validation". Beyond giving a good estimate of the generalization performance, an important benefit of this strategy is that it enables to study the variability of the performances. However, running many splits may be computationally expensive with models that are slow to train.

3.1.2. The need of an additional validation set

Often, it is useful to make choices on the model to maximize prediction performance: make changes on the architecture, tune hyper-parameters, perform early stopping... As the test-set performance is our best estimate of prediction performance, it would be be natural to run cross-validation and pick the best model. However, in such a situation, the performances reported on the testing set will be biased upwards: a data-dependent choice has been made on this test set. There are two main solutions to this issue. The first one is usually applied when the model training is fast and the dataset is of small size. It is called nested cross-validation. It consists in running two loops of cross-validation, one nested into the

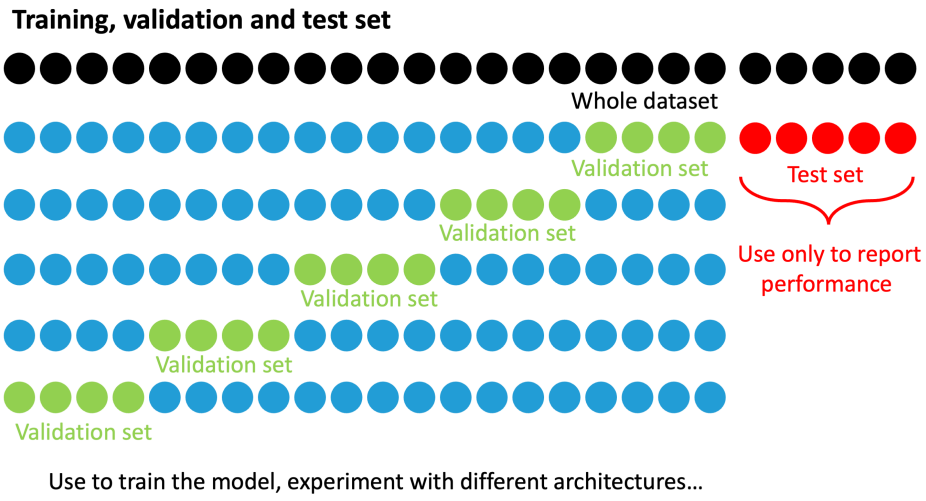


Figure 8: *A standard approach consists in splitting the whole dataset into training, validation and test sets. The test set must be isolated from the very beginning, left untouched until the end of the study and only be used to evaluate the performance. The training and validation sets are often used in a cross-validation manner. They can be used to experiment with different architectures and tune parameters.*

other. The inner loop serves for hyperparameter tuning or model selection while the outer loop is used to evaluate the performance. The second solution is to separate from the whole dataset the test set, which will only be used to evaluate the performances. Then, the remainder of the dataset can be further split into training data and data used to make modeling choices, called the validation set. Such a procedure is illustrated on Figure 8. Commonly, the training and validation sets will be used in a cross-validation manner. They can then be used to experiment with different models, tune parameters .... It is absolutely crucial that the test set is isolated at the very beginning, before any experiment is done. It should be left untouched and used only at the end of the study to report the performances. As for the split between training and validation set, it is desirable that stratification is done when isolating the test set.

If the dataset is very small, nested cross-validation should be preferred as it gives better testing power than hold-out: all the data are used alternatively for model testing. If the dataset feels too small to split in train, validation, test, it may be too small to conduct a trustworthy machine-learning study [8].

3.1.3. Various sources of data leakage

Data leakage denotes cases where some information from the training set has "leaked" into the test set. As a consequence, the estimation of the performances is likely to be optimistic. Data leakage can be introduced in many ways, some of which are particularly insidious and may not be obvious to a researcher that is not familiar with a specific application field. Below, we describe some common causes of data leakage. A summary can be found in Box 3.

A first basic cause of data leakage is to use the whole dataset for performing various operations on the data. A very common example is to perform feature selection using the whole dataset and then to use the selected features for model training. A similar situation is when dimensionality reduction is performed on the whole dataset. If this is done in an unsupervised manner (for example using principal component analysis), it is likely to introduce less bias in the performance estimation because the target is not used. It nevertheless remains, in principle, a bad practice. A common practice in deep learning is to perform early stopping, i.e. use the validation set to determine when to stop training. If this is the case, the validation performances can be overoptimistic and a separate test dataset should be used to report performance. Another cause of data leakage is when there are multiple longitudinal visits (i.e. the patient is evaluated at several time-points) or multiple modalities

for a given patient. In such as case, one should never put data from the same patient in both the training and validation sets. For instance, one should not, for a given patient, put the visit at month 0 in the training set and the visit at month 6 in the validation set. Similarly, one should not use the magnetic resonance imaging (MRI) data of a given patient for training and the positron emission tomography (PET) image for validation. A similar situation arises when dealing with 3D medical image. It is absolutely mandatory to avoid putting some of the 2D slices of a given patient in the training set and the rest of the slices in the validation set. More generally, in medical applications, the split between training and test set should always be done at the patient level. Unfortunately, data leakage is still prevalent in many machine learning studies on brain disorders. For instance, a literature review identified that up to 40% of studies on convolutional neural networks for automatic classification of Alzheimer's disease from T1-weighted MRI potentially suffered from data leakage [9].

Box 3: Some common causes of data leakage

- Perform feature selection using the whole dataset
- Perform dimensionality reduction using the whole dataset
- Perform parameter selection using the whole dataset
- Perform model or architecture search using the whole dataset
- Report the performance obtained on the validation set that was used to decide when to stop training (in deep learning)
- For a given patient, put some of its visits in the training set and some in the validation set
- For a given 3D medical image, put some 2D slices in the training set and some in the validation set

3.1.4. Statistical testing

Sources of variance. Train-test splits, cross-validation, and the like seek to estimate the expected generalization performance of a learning procedure. Keeping test data rigorously independent from algorithm development minimizes the bias of this estimation. However, the are

Table 1: *Accuracies obtained by different ML models on a binary classification task. Which model performs best? While it is quite likely that the convolutional neural network outperforms the two other models, it is less clear for the two other models. It seems that the support vector machine results in a slightly higher accuracy but is it due to random fluctuations in the benchmarks? Will the difference carry over to new data?*

Model	Accuracy
Logistic regression	0.72
Support vector machine	0.75
Convolutional neural network	0.95

multiple sources of arbitrary variations in these estimates. The most obvious one is the intrinsic randomness of certain aspects of learning procedures, such as the random initial weights in deep learning. Indeed, while fixing the seed of the random number generator may remove the randomness on a given train data, this stability is misleading given this choice is arbitrary and not representative of the overall behavior of the machine-learning algorithm on the data distribution of interest [10]. A systematic study of machine-learning benchmarks [11] shows that their most important sources of variance are:

Choice of test data / split. A given test set is an arbitrary sample of the actual population that we are trying to generalize to. As a result, the corresponding measure of performance is an imperfect estimate of the actual expected performance. Section 3.2, below, gives the resulting confidence intervals for a fixed test set. Using multiple splits, and thus multiple test sets, improves the estimation [11], though it makes computing confidence intervals hard [12].

Hyper-parameter optimization. The choice of hyper-parameters is imperfect, for instance because of limited resources to tune these hyper-parameters. Another attempt to tune hyper-parameter would lead to slightly different choice. Thus benchmarks do not give an absolute characterization of a learning procedure, but are muddied by imperfect hyper-parameters.

Random seeds. As mentioned above, random choices in a learning procedure –initial weights, random drop-out for neural networks or bootstraps in bagging– lead to uncontrolled fluctuations in benchmarking results that do not characterize the procedure's ability to generalize to new data.

Conclusions must account for benchmarking variance With all these sources of arbitrary variance, the question is: given benchmarks of a learning procedure performance, or improvement, is it likely to generalize reliably to new data or rather to be due to benchmarking fluctuations. Considering for instance the performance metrics in Table 1, it seems a safe bet to say that the convolutional neural network outperforms the two others but what about the difference between the two other models? From an application perspective, the question is whether this observed difference is likely to generalize to new data.

To answer this question, we must account for estimation error for the expected generalization performance from the different sources of uncontrolled variance in the benchmarks, as listed above. The first source of error comes from the limited sample size to test the predictions of the different learning procedures. Indeed, suppose that the testing set was composed of 100 samples. In that case, if only 3 more samples had been misclassified by the support vector machine, the two models would have had the same performance. A difference of 3 out of 100 could be easily due having drawn 3 samples not representative of the population. Other sources of variance are due to how stable the learning pipeline is: sensitivity to hyperparameters, random initialization...

A simple statistical testing procedure Training and testing a prediction pipeline multiple times is needed to estimate the variability of the performance measure. The simplest solution is to do this several times while varying the arbitrary factors, such as split between the train and the test or random initialization (see Box 4). The resulting set of performance measures is similar to bootstrap samples and can be used to draw conclusions on the distribution of performances in a test set. Confidence intervals can be computed using percentiles of this distribution. Two learning procedures can be compared by counting the number of times that one outperforms the other: outperforming 75% of the times is typically considered as a reliable improvement [11]. If the available computing power enables training learning procedures only a few times, empirical standard deviations should be used, as they require less runs to estimate. The improvements brought by a learning procedure can then be compared to these standard deviations.

Note these procedures do not perform classic null-hypothesis significance testing, which is difficult here. In particular the standard error across the various runs should not be used instead of the standard deviation: the standard error is the standard deviation divided by the number of runs. The number of runs can be made arbitrarily large given enough compute power, thus making the standard error arbitrarily small. But

Box 4: Statistical procedure to characterize a learner

1. Perform k runs of:

   (a) randomly splitting out a test set,

   (b) training the learning procedure on the train set,

   (c) and measuring the performance p on the test set.

   Choose different values of arbitrary parameters (such as random seeds) on each run, and if enough computing power, run hyper-parameter optimization each time. This results a set of performance measure $\mathcal{M} = \{m_1, ..., m_k\}$.

2. Use all the values $\{m_1, ..., m_k\}$ to conclude on the performance of the learner:

   Confidence intervals are given by percentiles of $\mathcal{M}$

   Standard deviation of $\mathcal{M}$ can be used to gauge typical variance of performance, as it requires performing a smaller number of runs k than percentiles. Standard error should not be used (see text)

   Learner comparison can be done by comparing two such set of values $\mathcal{M}$ and $\mathcal{M}'$, typically counting the fractions of values in $\mathcal{M}$ that outperform $\mathcal{M}'$ (without any pairing). Statistical procedures such as t-test should not be used (see text)

in no way does the uncertainty due to the limited test data vanish. This uncertainty can be quantified for a fixed test set –see 3.2, but in repeated splits or cross-validation it is difficult to derive confidence intervals because the runs are not independent [13, 12]. In particular, *it is invalid to use a standard hypothesis test –such as a T-test– across the different folds of a cross-validation.* There are some valid options to perform hypothesis testing in a cross-validation setting [14, 12], but they must be implemented with care.

Another reason not to rely on null-hypothesis testing is that their statistical significance only asserts that the expected performance –or improvement– is non zero over a test population of infinite size. From a practical perspective, we care about meaningful improvements on test sets of finite size, which is related to the notion of *acceptance* tests –as

opposed to *significance*– in the Neyman-Pearson framework of statistical testing [15]. Unlike null-hypothesis significance testing, it requires choosing a non-zero difference considered as acceptable, for instance as implicitly set by considering that a new learning procedure should improve upon an existing one 75% of the times –far from chance, which lies at 50%.

3.2 Generalization to an external population

The importance of external validation The procedures describe above characterize the expected error of a learning procedure applied on a given population. A related, but different, question is that of characterizing the error of a given predictive model, typically output by a training machine-learning procedure on a study population. That second question, related to the notion of *external validity*, is important for two reasons. First, it characterizes the specific predictive model that will be used in practice, "in production". Indeed, variance in the learning procedure will lead to arbitrary variation in model performance as large as typical improvements achieved by developing better models [11]. Second, characterizing the model on the *target* population may be important, as it may differ markedly from the study population. Indeed, the techniques in the previous section rely on splitting the initial dataset in training and testing (or validation) set; hence these different sets are by construction drawn from the same population, and have similar characteristics (data coming from the same hospital/centers/countries, similar age/sex . . .). They only demonstrate the ability of the model to generalize to new but similar data. To better assess model utility, guidelines on evaluating clinical prediction models, insist on external validation using data collected later in time, or in a different geographical area [16].

Testing whether a prediction model can generalize to dissimilar data is important as it is all too frequent that the study sample, on which the model was developed, does not represent the target population [17]. The target data may for instance come from different hospitals and different countries, be acquired with different acquisition devices and protocols or with different sociodemographic or clinical characteristics than those of the training data. For instance, it has been shown that the type of MRI scanner can have a substantial impact on the generalization ability of ML models. To assess such generalization ability, a common practice is to use one or several additional datasets for testing, these datasets being acquired using different protocols and at different sites (Figure 9). Most often, these datasets come from other research studies (different from the one used for training). However, research studies do not usu-

ally reflect well clinical routine data. Indeed, in research studies, the acquisition protocols are often standardized and rigorous data quality control is applied. Moreover, participants may not be representative of the target population. This can be due to inclusion/exclusion criteria (for instance excluding patients with vascular abnormalities in a study on Alzheimer's disease) or due to uncontrolled biases. For instance, participants to research studies tend to have a higher socioeconomic status than the general population. Therefore, it is highly valuable to also perform validation on clinical routine data, whenever possible, as it is more likely to reflect "real-life" situations. One should nevertheless be aware that a given clinical routine dataset may come with specificities that may not generalize to all settings. For instance data collected within a specialized center of a university hospital may substantially differ from that seen by a general practitioner.

Testing procedures for external validation External validation of a predictive model relies on an independent test set and not cross-validation. Statistical testing thus amount to derive confidence intervals or null-hypothesis significance testing for the metric of interest on this test set, exactly as when characterizing a diagnostic test [18].

For simple metrics that rely on counting successes, such as accuracy, sensitivity, PPV, NPV, the sampling distribution can be deduced from a binomial law. Table 2 gives such confidence intervals for different set of the test set and different values of the ground-truth accuracy. These can be easily adapted to other counts of errors as follows:

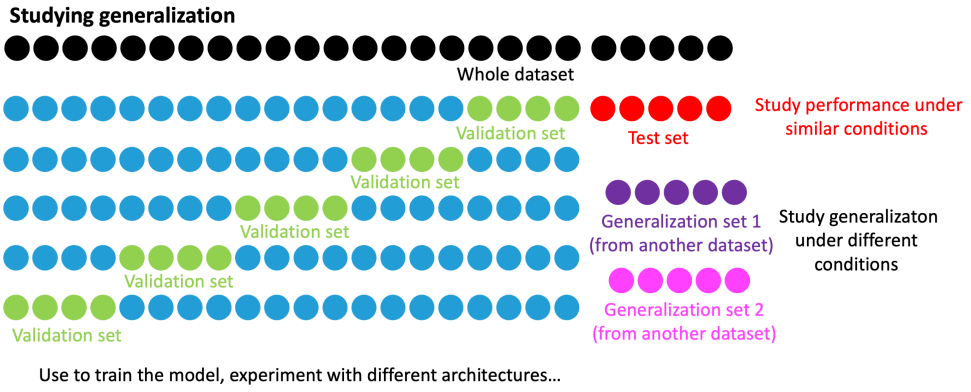


Figure 9: *In order to assess the generalization ability of a model under different conditions (such as data coming from different hospitals/countries, acquired with different devices and protocols...), a common practice is to use one or several additional datasets that come from other studies than the one used for training.*

Table 2: ***Binomial confidence intervals on accuracy*** *(95% CI) for different values of ground-truth accuracy.*

N	65%	80%	90%	95%
100	[-9.0% 9.0%]	[-8.0% 8.0%]	[-6.0% 5.0%]	[-5.0% 4.0%]
1000	[-3.0% 2.9%]	[-2.5% 2.4%]	[-1.9% 1.8%]	[-1.4% 1.3%]
10000	[-0.9% 0.9%]	[-0.8% 0.8%]	[-0.6% 0.6%]	[-0.4% 0.4%]
100000	[-0.3% 0.3%]	[-0.2% 0.2%]	[-0.2% 0.2%]	[-0.1% 0.1%]

Accuracy	N is the size of the test set
Sensitivity	N is the number of negative samples in the test set
PPV	N is the number of positively classified test samples
NPV	N is the number of negatively classified test samples

We believe it is very important to have in mind the typical orders of magnitude reported in Table 2. It is not uncommon to find medical classification studies where the test set size is about a hundred or less. In such a situation, the uncertainty on the estimation of the performance is very high.

These parametric confidence intervals are easy to compute and refer to. But actual confidence intervals may be wider if the samples are not i.i.d. In addition, some interesting metrics, such as AUC ROC, do not come with such parametric confidence interval. A general and good option, applicable to all situations, is to approximate the sampling distribution of the metric of interest by bootstrapping the test set.

Finally, note that all these confidence intervals assume that the available labels are the ground truth. In practice, medical truth is difficult to establish, and label error may bias the estimation of error rates.

When comparing two classifiers, a McNemar's test is useful to test whether the observed difference in errors can be explained solely by sampling noise [19, 20]. The test is based on the number of samples misclassified by one classifier and not the other, n_{01} and vice versa n_{10}. The test statistics is then written $(|n_{01} - n_{10}| - 1)^2/(n_{01} + n_{10})$; it is distributed under the null as a χ^2 with 1 degree of freedom. To compare classifiers' scanning the tradeoff between specificity and sensitivity without choosing a specific threshold on their score one option is to compare areas under the curve of the ROC, using the DeLong test [21] or a permutation scheme to define the null [22].

4. Conclusion

Evaluating machine learning models is crucial. Can we claim that a new model outperforms an existing one? Is a given model trustworthy enough to be "deployed", making decisions in actual clinical settings? A good answer to these questions requires model-evaluation experiments adapted to the application settings. There is no one-size-fits-all solution. Multiple performance metrics are often important, chosen to reflect target population and cost-benefit trade-offs of decisions, as discussed in section 2. The prediction model must *always* be evaluated on unseen "test" data but different evaluation goals lead to procedures to choose this test data. Evaluating a "learner" –a model-construction algorithm– leads to cross-validation, while evaluating the fitness of a given prediction rule –as output by model fitting– calls for left-out data representative of the target population. In all settings, accounting for uncertainty or variance of the performance estimate is important, for instance to avoid investing in models that bring no reliable improvements.

Acknowledgments

This work was supported the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute), ANR-10-IAIHU-06 (Agence Nationale de la Recherche-10-IA Institut Hospitalo-Universitaire-6), ANR-20-CHIA-0026 (LearnI), and ANR-17-CE23-0018 (DirtyData).

References

[1] Pedregosa F, et al (2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12(85):2825–2830

[2] Powers D (2011) Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. Journal of Machine Learning Technologies 2(1):37–63

[3] Naeini MP, Cooper G, Hauskrecht M (2015) Obtaining well calibrated probabilities using bayesian binning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence

[4] Vickers AJ, Van Calster B, Steyerberg EW (2016) Net benefit approaches to the evaluation of prediction models, molecular markers, and diagnostic tests. bmj 352

[5] Poldrack RA, Huckins G, Varoquaux G (2020) Establishment of best practices for evidence for prediction: a review. JAMA psychiatry 77(5):534–540

[6] Barocas S, Hardt M, Narayanan A (2019) Fairness and Machine Learning. fairmlbook.org, http://www.fairmlbook.org

[7] Varoquaux G, Raamana PR, Engemann DA, Hoyos-Idrobo A, Schwartz Y, Thirion B (2017) Assessing and tuning brain decoders: cross-validation, caveats, and guidelines. NeuroImage 145:166–179

[8] Varoquaux G (2018) Cross-validation failure: Small sample sizes lead to large error bars. Neuroimage 180:68–77

[9] Wen J, Thibeau-Sutre E, Diaz-Melo M, Samper-González J, Routier A, Bottani S, Dormont D, Durrleman S, Burgos N, Colliot O, et al (2020) Convolutional neural networks for classification of alzheimer's disease: Overview and reproducible evaluation. Medical image analysis 63:101694

[10] Bouthillier X, Laurent C, Vincent P (2019) Unreproducible research is reproducible. In: International Conference on Machine Learning, PMLR, pp 725–734

[11] Bouthillier X, Delaunay P, Bronzi M, Trofimov A, Nichyporuk B, Szeto J, Mohammadi Sepahvand N, Raff E, Madan K, Voleti V, et al (2021) Accounting for variance in machine learning benchmarks. Proceedings of Machine Learning and Systems 3:747–769

[12] Bates S, Hastie T, Tibshirani R (2021) Cross-validation: what does it estimate and how well does it do it? arXiv preprint arXiv:210400673

[13] Bengio Y, Grandvalet Y (2004) No unbiased estimator of the variance of k-fold cross-validation. Journal of machine learning research 5(Sep):1089–1105

[14] Nadeau C, Bengio Y (2003) Inference for the generalization error. Machine learning 52(3):239–281

[15] Perezgonzalez JD (2015) Fisher, Neyman-Pearson or NHST? A tutorial for teaching data testing. Frontiers in psychology p 223

[16] Moons KG, Altman DG, Reitsma JB, Ioannidis JP, Macaskill P, Steyerberg EW, Vickers AJ, Ransohoff DF, Collins GS (2015) Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (tripod): explanation and elaboration. Annals of internal medicine 162(1):W1–W73

[17] Dockès J, Varoquaux G, Poline JB (2021) Preventing dataset shift from breaking machine-learning biomarkers. GigaScience 10(9):giab055

[18] Shapiro DE (1999) The interpretation of diagnostic tests. Statistical methods in medical research 8(2):113–134

[19] Leisenring W, Pepe MS, Longton G (1997) A marginal regression modelling framework for evaluating medical diagnostic tests. Statistics in medicine 16(11):1263–1281

[20] Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation 10(7):1895–1923

[21] DeLong ER, DeLong DM, Clarke-Pearson DL (1988) Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics pp 837–845

[22] Bandos AI, Rockette HE, Gur D (2005) A permutation test sensitive to differences in areas for comparing roc curves from a paired design. Statistics in medicine 24(18):2873–2893

A. Appendix

A.1 Odds ratio and diagnostic-tests evaluation

Odds and odds ratio are frequently used in biostatistics and epidemiology, but less in machine learning. Here we give a quick introduction to these topics.

A.1.1. Odds

Odds are a measure of likelihood of an outcome: the ratio of the number of events that produce that outcome to the number that do not. The odds $O(a)$ of an outcome a are simply related to the probability $P(a)$ of

this outcome:

Odds of a
$$O(a) = \frac{P(a)}{1 - P(a)} \tag{1}$$

In other words, $O(a)$ is the number of times the event a would occur for each occurrence of the opposite event. This intuitive explanation has led odds to be often used for sports gambling. For instance, if the odds are 3 (or more specifically in gambling terminology 3 : 1) for FC Barcelona vs Real Madrid, it means that FC Barcelona has a probability of winning against Real Madrid of 75% ($P(a) = \frac{O(a)}{O(a)+1}$). Coming back to diseases, supposing that only a minority of the population is affected, if the odds of the disease are 1%, which can be written as 1 : 100, this means that for every diseased person in the population, there are 100 persons without it. The prevalence is thus $\frac{1}{101} = 0.99\% \approx 1\%$. One can see that when the prevalence is low, it is close to the odds, which is not the case when prevalence gets higher. This is true in general of probabilities and odds: when the probability is low, it is close to the odds.

A.1.2. Odds ratio and invariance to sampling

The odds ratio measures the association between two events, a and b, which we can arbitrarily call respectively outcome and property. The odds ratio is defined as the ratio of the odds of the outcome in the group where the property holds to that in the group where the property does not hold:

Odds ratio between a and b
$$OR(a, b) = \frac{O(a|b = +)}{O(a|b = -)} \tag{2}$$

To compute the odds ratio, the problem is fully specified by the counts in the following contingency table:

		Outcome a	
		$a+$	$a-$
Property b	$b+$	n_{++}	n_{-+}
	$b-$	n_{+-}	n_{--}

(3)

The odds are written: $O(a|b = +) = \frac{n_{++}}{n_{-+}}$ and $O(a|b = -) = \frac{n_{+-}}{n_{--}}$ hence the odds ratio reads

$$OR(a, b) = \frac{n_{++}}{n_{-+}} \frac{n_{--}}{n_{+-}}. \tag{4}$$

Note that this expression is unchanged swapping the role of a and b; the odds ratio is symmetric, $OR(a, b) = OR(b, a)$

Invariance to sampling Suppose we have sampled the population selecting with a frequency f on the outcome $a+$, for instance to oversample the positive outcome or the positive property [1]. In eq. 4, n_{++} is replaced by $f\,n_{++}$ and n_{+-} by $f\,n_{+-}$; however, the factor f cancels out and the overall expression of the odds ratio is unchanged. This is a central property of the odds ratio:

> **The odds ratio is unchanged by sample selection bias on one of the variables (a or b).**

This property is one reason why odds and odds ratio are so central to biostatistics and epidemiology: sampling or recruitment bias are an important concern in these fields. For instance, a case-control study has a very different prevalence as the target population, where the frequency of the disease is typically very low.

Confusion with risk ratio The odds ratio is often wrongly interpreted as a risk ratio –or relative risk–, which is more easily understood.

The risk ratio is the ratio of the probability of an outcome in a group where the property holds to the probability of this outcome in a group where this property does not hold. The risk ratio thus differs from the odds ratio in that it is expressed for probabilities and not odds. Even though the values for odds ratio and risk ratio are often close because, in most diseases being diseased in much less likely than not, they are fundamentally different because the odds ratio does not depend on sampling whereas the risk ratio does.

A.1.3. Likelihood ratio of diagnostic tests or classifiers

The likelihood ratio used to characterize diagnostic tests or classifiers is strongly related to the odds ratio introduced above, though it is not strictly speaking an odds ratio. It is defined as:

$$\mathrm{LR}+ = \frac{P(T+\,|D+)}{P(T+\,|D-)} \tag{5}$$

Using the expressions in Box 1 and the fact that $P(T+|D+) = 1-P(T-|D+)$, the LR+ can be written as:

$$\mathrm{LR}+ = \frac{\text{Sensitivity}}{1-\text{Specificity}} \tag{6}$$

[1] Indeed, thankfully, many diseases have a prevalence much lower than 50%, e.g. 1% which is already considered a frequent disease. Therefore, in order to have a sufficient number of diseased individuals in the sample without dramatically increasing the cost of the study, diseased participants will be oversampled. One extreme example, but very common in medical research, is a case-control study where the number of diseased and healthy individuals is equal.

Link to pre-test and post-test odds We can write this in terms of the contingency table in eq. 3 (the link to the confusion matrix in Figure 1 is given by $a = D$, $b = T$ and thus $n_{++} = TP$, $n_{-+} = FP$, $n_{+-} = FN$, $n_{--} = TN$):

$$\text{LR+} = \frac{n_{++}}{n_{++} + n_{+-}} \frac{n_{-+} + n_{--}}{n_{-+}} \tag{7}$$

$$= \underbrace{\frac{n_{++}}{n_{-+}}}_{\frac{P(D+|T+)}{P(D-|T+)} = O(D+|T+)} \underbrace{\frac{n_{-+} + n_{--}}{n_{++} + n_{+-}}}_{\frac{P(D-)}{P(D+)} = \frac{1}{O(D+)}} \tag{8}$$

$$\text{LR+} = \frac{O(D + |T+)}{O(D+)} \tag{9}$$

Indeed, $O(D + |T+) = \frac{P(D+|T+)}{1-P(D+|T+)} = \frac{P(D+|T+)}{P(D-|T+)}$ and $O(D+) = \frac{P(D+)}{1-P(D+)} = \frac{P(D+)}{P(D-)}$.

$O(D+)$ is called the pre-test odds (the odds of having the disease in the absence of test information). $O(D+|T+)$ is called the post-test odds (the odds of having the disease once the test result is known).

Equation 9 shows how the LR+ relates pre- and post-test odds, an important aspect of its practical interpretation.

Invariance to prevalence If the prevalence of the population changes, the quantities are changed as follows: $n_{++} \to f\, n_{++}$, $n_{+-} \to f\, n_{+-}$, $n_{-+} \to (1-f)\, n_{-+}$, $n_{--} \to (1-f)\, n_{--}$, affecting LR+ as follows:

$$\text{LR+} = \frac{f\, n_{++}}{(1-f)\, n_{-+}} \frac{(1-f)\, n_{-+} + (1-f)\, n_{--}}{f\, n_{++} + f\, n_{+-}}. \tag{10}$$

The factors f and $(1-f)$ cancel out, and thus the expression of $LR+$ is unchanged for a change of the pre-test frequency of the label (prevalence of the test population). This is alike odds ratios, though the likelihood ratio is not an odds ratio (and does not share all properties; for instance it is not symmetric).