Quantitative and Computational Methods in Behavioral Sciences

PsychOpen GOLD

Original Article

Check for updates

Independent Validation as a Validation Method for Classification

Tina Braun [1], Hannes Eckert [2], Timo von Oertzen [3]

[1] *Charlotte-Fresenius University, Wiesbaden, Germany.* **[2]** *University of the Bundeswehr, Munich, Germany.* **[3]** *Max Planck Institute for Human Development, Berlin, Germany.*

Quantitative and Computational Methods in Behavioral Sciences, 2023, Article e12069, https://doi.org/10.5964/qcmb.12069

Received: 2023-05-30 • **Accepted:** 2023-08-03 • **Published (VoR):** 2023-12-22

Handling Editor: Timothy Brick, Pennsylvania State University, University Park, PA, USA

Corresponding Author: Tina Braun, Charlotte-Fresenius University, Moritzstrße 17a, 65185, Wiesbaden, Germany. E-mail: tina.braun@charlotte-fresenius-uni.de

Supplementary Materials: Code, Data [see Index of Supplementary Materials]

Abstract

The use of classifiers provides an alternative to conventional statistical methods. This involves using the accuracy with which data is correctly assigned to a given group by the classifier to apply tests to compare the performance of classifiers. The conventional validation methods for determining the accuracy of classifiers have the disadvantage that the distribution of correct classifications does not follow any known distribution, and therefore, the application of statistical tests is problematic. Independent validation circumvents this problem and allows the use of binomial tests to assess the performance of classifiers. However, independent validation accuracy is subject to bias for small training datasets. The present study shows that a hyperbolic function can be used to estimate the loss in classifier accuracy for independent validation. This function is used to develop three new methods to estimate the classifier accuracy for small training sets more precisely. These methods are compared to two existing methods in a simulation study. The results indicate overall small errors in the estimation of classifier accuracy and indicate that independent validation can be used with small samples. A least square estimation approach seems best suited to estimate the classifier accuracy.

Keywords

independent validation, classifiers, classifier accuracy, simulation

Independent Validation as a Validation Method for Classification

The study of group differences is of particular importance in all of the empirical sciences, as indicated by a high number of publications (Wee, 2000; Weisberg et al., 2011; Zhao et al., 2020). For example, Counsell and Harlow (2017) showed that nearly 40% of all analyses included a univariate comparison of means, and ANOVAs as well as *z*- and *t*-tests were among the most popular methods. Both parametric and nonparametric tests make certain assumptions (Beins & McCarthy, 2019; Kim & von Oertzen, 2018), which can be a potential source of error if not fulfilled. For situations where these assumptions cannot be met, the usage of classifiers has been proposed as an alternative (Kim & von Oertzen, 2018).

In this approach, a classifier is trained to identify the group of a participant based on the data. If both groups do not differ, no classifier is able to perform better than guessing on this task. A performance above guessing ensures that the groups differ. For an illustration, assume a researcher wants to identify how strongly smokers and non-smokers differ in a reaction task, where the result of the reaction task is of an unknown distribution. They set up this question as a task to identify whether a person is a smoker or a non-smoker based on the reaction task result. If a classifier is able to detect smokers better than by just guessing the more frequent group, this indicates that the outcome's distribution differs between smokers and non-smoker. The approach and its requirements will be described in more detail in the next section.

Testing group differences by classifiers requires to estimate the accuracy of a classifier. Multiple methods exist for this estimation. However, for conventional methods, the distribution of correct classifications does not follow any known distribution. This prevents the unbiased use of binomial tests. To circumvent this problem, Independent Validation (IV) was suggested (Kim & von Oertzen, 2018). An advantage of IV is that the accuracy of the classifier is unbiased through dependencies within the data. However, IV is a process in which the accuracy, with which new data points are classified, changes. This will produce some estimation bias for low *N*. For this purpose, this article will begin by revisiting how to use classifiers to analyze group differences without known group distributions and briefly sketch the associated complications. Then, IV will be revisited more closely, in particular the expected accuracy for small training set sizes. The aim of the present study is to investigate how small training set sizes can be used in the IV process and to compare different methods to estimate the classifier IV-accuracy.

Using Classifiers to Analyze Group Differences

One problem with using classical methods to study group differences is that it is frequently not reported whether the underlying assumptions have been tested (Counsell & Harlow, 2017; Hoekstra et al., 2012). Regardless of whether the assumptions have not

been tested or were merely not reported, the transparency and replicability of research is limited due to transparency issues. An alternative method for studying group differences that does not require testing assumptions would, hence, be advantageous. Such a method has been proposed in the shape of using classifier vectors (Bay & Pazzani, 2001; Kim & von Oertzen, 2018). Most classifiers do not require any model assumptions concerning the distribution of the data. This makes them particularly suited to investigate situations where no model of the data is available, as is, for example, often the case with big data (Chen et al., 2012). Examples for classifiers are *k*-NN classifier, support vector machines (Cortes & Vapnik, 1995), random decision forests (Ho, 1995), and decision trees (Quinlan, 1986). Different classifiers are differently well suited to varying types of data sets. Choosing a classifier suited to one's data increases the chances to find group differences present in the data. A wide range of literature already addresses this problem (e.g., Bishop, 2006). In addition, a suitable method is required to estimate the classifier's accuracy. The current article focuses on this second requirement.

As classifiers are used to detect group differences, the Linear Discriminant Analysis (LDA) is chosen in this article. LDA will be introduced in the next subsection. However, the estimation of the accuracy is analogous for most other classifiers.

To revisit the smoker vs. non-smoker example, the researcher uses an LDA on their data to train it on separating smokers from non-smokers. Using the methods introduced in this article, they then estimate the probability that a new participant for whom only the reaction task data is known is correctly classified as smoker or non-smoker. If this probability exceeds the probability one would have by just picking the larger group, this indicates that smokers and non-smokers differ in the reaction task.

Linear Discriminant Analysis

LDA was introduced in its original form by Fisher (1936) as a method aimed at finding variables that best distinguish the data of groups (Boedeker & Kearns, 2019). Later, the method was developed further to enable predicting group membership (Boedeker & Kearns, 2019; Huberty & Hussein, 2003). For a data set with two groups and a number of continuous variables, a training data set is formed from both groups. This training data set is used to calculate the joint estimated covariance matrix and mean for each group. Then, with respect to the joint covariance matrix, the distances of a new data point to the respective estimated means of both groups are calculated. The new item is assigned to the class to which the distance is smaller. This constitutes a maximum likelihood approach, as the item is assigned to the group with the higher likelihood. It is apparent that LDA is more complex than other classifiers like k-NN but performs better in many situations (Boedeker & Kearns, 2019).

Validation of Classifiers

Classifiers can classify new elements more or less reliably. The accuracy of a classifier is a performance indicator to quantify this reliability. In the example above, the accuracy would be the probability to detect a smoker or non-smoker correctly. The measure indicates the probability with which an element drawn randomly from the population is correctly classified (Kohavi, 1995). The optimal accuracy is usually not known. However, with the help of test runs, the accuracy can be estimated. This estimation is typically performed by computing the percentage of correctly classified elements out of the total number of classified elements. Therefore, information on the actual affiliation of the elements is necessary, meaning that the group membership needs to be known.

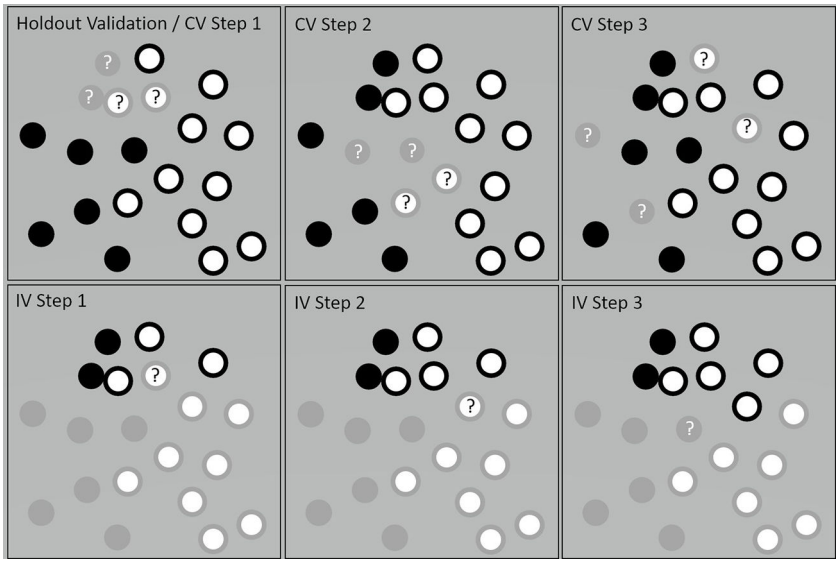
If there are two equally sized groups in a study situation, but the data does not differ between the groups, the classifier cannot detect a pattern that associates the expressions with the respective group membership. Thus, it would only guess the group of new elements with an accuracy of 50%. Consequently, if a classifier assigns new items better than a coin toss would, there must be differences between the groups (Kim & von Oertzen, 2018). Accordingly, when examining group differences, it is only by comparing the accuracy of the classifier with the accuracy of a coin toss classifier that it is possible to assess whether differences exist between groups. Therefore, the classifier accuracy is of central importance in the study of group differences using classifiers.

In the past, various validation procedures have been introduced to determine the classifier accuracy. The basic problem of all validation methods is primarily that in the practical application usually only a limited sample size is available. This limited data set must then be used to train the classifier. At the same time, however, the classifier accuracy must also be estimated on the same data set (Sahiner et al., 2008). To ensure independence between the tests of the tested items, it must be avoided to evaluate the performance of the classifier on elements that are simultaneously included in the training data set.

The simplest method to address this problem is the holdout or test set method, illustrated in a simple example in the upper left panel of Figure 1. In this method, the data set is divided into two mutually exclusive subsamples: the training data set and the test data set (Kohavi, 1995; Santafe et al., 2015). The training data set is used exclusively for the learning phase of the validation, while the classifier accuracy is estimated on the test data set. In the example above, the points would be participants with some properties describing the reaction task on the horizontal and vertical axes. The classifier is trained on the training set (solid points in Figure 1), and its performance is then estimated on the holdout set (gray points in Figure 1).

Figure 1

Illustration of Cross Validation vs. Independent Validation



Note. Filled and empty circles represent data points of two classes. In holdout validation (upper left panel), the gray data points are used for testing the classifier trained on all non-gray data points. In CV (upper panels), folds (here of four data points) are used to test the classifier trained on all other data points similar to the holdout validation, but afterwards, the next fold of same size is used for testing, again using all other points as training set. LOO validation is a special case of CV where the folds contain only one data point. In IV (lower panels), an initial set (non-gray points) is used for training, then one point of the remaining data points is tested. In the next step, this point is added to the training set, and a new point is tested (2nd lower panel), which is then repeated for all other points (3rd lower panel). This way, no data point is tested which has been used before to train the classifier, ensuring independence of the tests.

Although the holdout method is one of the simplest methods, it bears some problems. For instance, the performance of the classifier suffers from a small training data set as the already limited data set is further subdivided (Kim & von Oertzen, 2018). According to Santafe et al. (2015), the holdout estimator overestimates the error of the classifier, resulting in an underestimation of the accuracy.

More common is the k-fold cross-validation method (CV; Kohavi, 1995). The three upper panels in Figure 1 illustrate three steps in CV. In this method, the data is randomly divided into k folds of equal size. The classifier is then trained on $k - 1$ folds, and finally performance is tested on the one remaining fold. This process is repeated for all k folds, resulting in k classification models and thus as many estimates of the classifier's performance. For the estimation of the accuracy, the mean value of k individual estimates is calculated (Santafe et al., 2015). According to Kim and von Oertzen (2018), CV is

characterized by higher statistical power at higher numbers of folds. With more folds, the size of the actual training data set per run increases. This allows the classifier to learn the patterns in the data more effectively. Regardless of the choice of the training data set size per run, the total number of tested items remains the same across all runs, and the training set in each run remains as large as possible as it includes $\frac{k-1}{k}N$ elements of the data set when N is the total size of the data set. However, CV is also subject to bias, albeit reduced compared to the holdout method (Santafe et al., 2015). As the number of folds in CV increases (i.e., a larger training data set per run), the variance of the performance estimator tends to increase. If the number of folds is maximized (i.e., $k = N$), this represents a special case of CV and is referred to as the Leave One Out (LOO) method.

Lanka et al. (2020) conclude from the results of previous work that the accuracy of classifiers using CV, while considered high on the training set, reduces dramatically when the classifier is applied to a new sample (i.e., a sample not previously used in the CV process). At the same time, Bouckaert (2003) points out that dependencies arise in the process of CV as a result of repeated use of the data. In CV, it is true that none of the elements of the data set are used at the same time to train and test the classifier, which would violate the requirement of independence. Nevertheless, all elements are used multiple times to train the classifier, which violates the independence assumptions of most hypothesis tests.

If the data is not independent, CV accuracy is too optimistic and overestimates the actual accuracy, consequently leading to a too liberal significance test (Kohavi, 1995). In contrast, if the data is independent, it is commonly assumed that the CV accuracy is binomially distributed (see Salzberg, 1997). Several studies have shown that this assumption is incorrect. In fact, Salzberg (1997) himself notes that while all test sets are independent in k-fold CV, the training sets are clearly not due to the overlaps in the process of CV. According to Dietterich (1998) and Bouckaert (2003), alpha error accumulation occurs when CV is used in practice. To eliminate the dependencies in both testing and learning phases, Kim and von Oertzen (2018) propose IV.

Independent Validation

With this new method, the accuracy is measured only with elements that were not already included in previous steps for training the classifier (see the three lower panels of Figure 1). For this purpose, the authors establish the minimal condition by which the estimates of the accuracies are independent. Accordingly, no element that is to be classified in the test run may have previously been part of a training set during the validation process. If this condition is met, the classifications in the test runs are independent of one another. Since the resulting independent classification results from the test runs are the basis for estimating the accuracy of the classifier, the independence requirement is thus

satisfied. Therefore, the estimated IV accuracy can be used in significance tests to find group differences or to compare different classifiers with each other.

The IV process is logically constructed according to the minimal condition. First, the entire data set is divided into a small initial training set and a test set. The test set also has the function of a residual pool. For both subsets, the data is drawn randomly, with only the number of elements predetermined. After the initial split, one or more data points are randomly drawn from the test data set and classified by the classifier. After this first test run, the tested elements are removed from the test data set and assigned to the training data set instead. The classifier is then trained again, including the new elements that have already been tested. This process is repeated with new single or multiple elements and continues until all elements of the test set have been used. Hence, elements are only transferred from the test set to the training set but not vice versa.

In the smoker example, the data set would be separated in a small group of participants for which it is known whether they smoke. The classifier is trained on those and then asked to classify one of the remaining participants. The classifier outcome of smoker or non-smoker is then compared to the true smoking status of that participant. Since this participant now has been tested, they are added to the training pool in the next step. Now the classifier trains with a training set one participant larger than before. Again, a new participant not in the training set is used to check whether the classifier is able to detect them as smoker or not, and then the participant is again added to the training set.

The estimated accuracy of IV is binomially distributed and can therefore be used for statistical testing (Kim & von Oertzen, 2018). The qualitative properties of IV correspond to those of CV. When comparing IV with the LOO method, it is noticeable that the final size of the training set is smaller for IV. Therefore, according to Kim and von Oertzen (2018), the estimated accuracy of IV is lower than that of the LOO method for small sample sizes. However, unlike LOO or CV in general, IV offers the advantage of independent estimates of classifier accuracy in the different runs. Compared to the conventional holdout method, IV shows a much higher power (Kim & von Oertzen, 2018).

In summary, IV has an advantage over CV and LOO methods in the applicability of statistical tests. At the same time, IV is a reasonable alternative to the holdout method due to its higher statistical power. This had previously been the only method that allowed valid use of tests to compare the performances of two classifiers (Kim & von Oertzen, 2018).

Formal Definitions

Building on Bax (2012), Boucheron et al. (2005), and Kim and von Oertzen (2018), the following definitions are made: Let $\Omega = \{(x, G)\} = \mathbb{R}^K \times \{-1, 1\}$ be a population. The elements of the population are pairs of K traits $x \in \mathbb{R}^K$ and a dichotomous class varia-

ble (groups) $G \in \{-1, 1\}$. In this study, we will restrict ourselves to the case $K = 1$ for simplicity; however, all results transfer to high-dimensional spaces. The data set $D = \{(x_1, G_1), ..., (x_N, G_N)\} \subseteq \Omega$ with size N is randomly drawn from the population. Let $C_{D_x} \in \{0, 1\}$ be the classification result of element x by a classifier. The classification result is false if $C_{D_x} \neq G_x$. Let the random variable $p_n = P(C_D(x) = G)$ be the estimated accuracy of the classifier for the population given a randomly drawn training data set of size n, where $p = lim_{n \to \infty} p_n$ is the maximum possible accuracy of the classifier for the population given sufficiently large training sets. Accordingly, p_n is the estimated accuracy for a training data set of size n.

According to Kim and von Oertzen (2018), the following formal definition of IV are used: Let C_{D_i} be a classifier trained with $D_i = \{(x_1, G_1), ..., (x_{i-1}, G_{i-1})\} \subseteq D$, that is, a subsample of $D \subseteq \Omega$, and let $M \leq N$ be a fixed number of tests. Let $q = \frac{1}{M} \sum_{i=N-M}^{N} \delta(C_{D_i}, Gi)$, then q is denoted as M-IV of the classifier. As explained above, an initial training set of size $n = N - M$ is chosen, and the correctness of the ith classification is tested. Then, the ith element is added to the training set. The classifier is trained again with the new training data set, and the process is repeated with the element $(i + 1)$. This sequence is repeated in the following validation process.

Remaining Issues Concerning Independent Validation

Some aspects of applicability are still unresolved in IV. The authors themselves emphasize that the choice of the initial training set size is a critical decision and has not yet been clarified (Kim & von Oertzen, 2018). According to the authors, this decision is comparable to both the selection of the number of folds in the k-fold CV and the chosen size ratio of the subsamples in the holdout method. The classifier cannot be trained sufficiently in the first runs of the IV process if the initial training data set is too small. If, on the other hand, the initial training data set is chosen too large, there are consequently too few elements in the test data set to be able to reliably estimate the classifier accuracy. The authors suggest an initial size of the training data set of about 20 elements as a guide for sufficiently large data sets. At the same time, they point out that the optimal choice of the initial training set size should be investigated in the future, which will be addressed in the present study.

A second remaining issue concerns the calculation of the accuracy in the context of IV. In practice, the result of IV is only a binary string, that is $C_{D_i} = 1$ or $C_{D_i} = 0$ due to correct or incorrect classifications. The simplest way to convert these values into an estimated accuracy is to calculate the percentage of correct classifications out of the total number of completed classifications. However, for a small number of elements in the initial training set, the performance of the classifier in the first runs is negatively biased. The reason for this is the small size of the training data set in the first runs, which causes the classifier to be insufficiently trained. Using the method of calculating the percentage

of correct classifications, each individual classification result is weighted equally. Hence, if all classification results are equally weighted, the accuracy tends to be underestimated.

The present article addresses both these points in a single new method that allows to estimate the population accuracy of the classifier even for very small *N*. The method starts the initial training set at very small numbers and subsequently corrects for the bias by estimating the functional form of the bias in dependency of the training set size. Two simulations are used to verify these results: The first simulation confirms the functional form of the bias, and the second simulation compares the performance of different estimation techniques for the function form. In result, it is possible to estimate the classifier accuracy even for very small *N* in an unbiased and independent form which allows for frequentist or Baysian tests.

Development of a Function Estimating the Accuracy Bias

In the following section a function is developed that approximates the accuracy bias. Let two groups $G \in \{-1, 1\}$ be given in a unidimensional feature space in which the probability densities of the features of the two groups are described by f_{-1} and f_1. Assume further that the groups are classified by a decision point. This situation is presented in Figure 2A. The ideal separation in the case of the mapped distributions is the point of the intersection $f_{-1}(x) = f_1(x)$ of the distribution functions. Without loss of generality, we assume that this point is at $x = 0$.

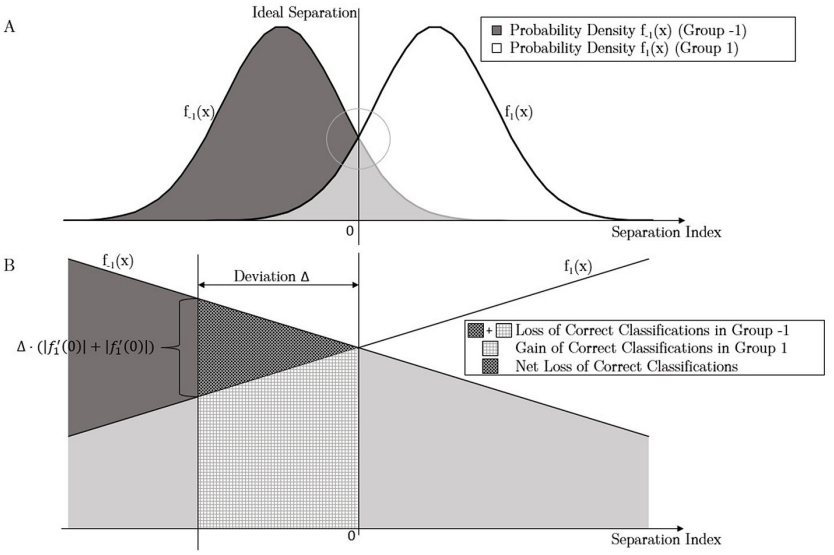
Assume further that a classifier is trained with a training data set of size n and estimates the separation point with standard error $SE = \frac{SD}{\sqrt{n}}$. Here, the SD is a fixed constant corresponding to the standard deviation of the estimator for a single element. Figure 2B shows the area around the intersection of the distributions enlarged. Again, the vertical line at zero corresponds to the ideal separation point. The perpendicular line mapped to the left of the ideal by distance Δ represents the separation point estimated by the classifier. Δ thus is the error of the estimate for the optimal separation point. The error originates from the sampling error in the subsample used by the classifier as the training data set.

Asymptotically, the curves in the area near the ideal separation point are approximately straight. In the following, $f_{-1}(x)$ and $f_1(x)$ in Figure 2B are therefore treated as straight lines within the depicted sector. The slopes of the curves can be calculated using the derivatives of the distribution functions. Thus, the value of $f'_{-1}(0)$ corresponds to the slope of f_{-1} close to zero, and the value of $f'_1(0)$ corresponds to the slope of f_1 close to zero.

Using the estimated separation point of the classifier with the deviation Δ and the ideal separation point results in areas between the two distribution curves in the range Δ. The area under f_{-1} between the ideal and the estimated separation is the loss of

Figure 2

Deduction to Approximate the Classifier Accuracy Bias



Note. Part A shows the distributions of both groups. The light gray area depicts the probability of incorrect classifications. The encircled area is enlarged in part B. Depicted is the deviation Δ of the classifier from the ideal separation point.

probability for correct classifications in group $G = -1$ due to the estimation error Δ. Thus, more elements from group $G = -1$ are classified by the classifier as belonging to the class $G = 1$ due to the estimated separation in Figure 2B. Simultaneously, more elements of group $G = 1$ are classified correctly. This additional probability of obtaining correctly classified elements from group $G = 1$ is the area under f_1. Thus, there is a simultaneous gain in one group and a loss of correct classifications in the other. The loss is always higher than the gain. Therefore, the area of the triangle between f_{-1} and f_1 is the net loss of correct classifications and accuracy, respectively. The methods suggested here aim at correcting this loss.

The calculation of the area of interest is possible by the formula $A = \frac{1}{2} \cdot g \cdot h$, since due to the simplifying assumptions it is assumed that the region is a triangle. Therefore, the height of the triangle is given by the deviation Δ. The base side of the triangle can also be calculated: One part of the base side is given by the value by which f_{-1} decreases in the range between the ideal and the estimated separation. For the second part of the base side, the same is true for the slope of f_1. Since the estimated cut can also be to the right of the ideal separation, absolute values are calculated to ensure generalizability. The

absolute values by which f_{-1} and f_1 increase and decrease, respectively, in the relevant range are calculated by $\Delta \cdot f'_{-1}(0)$ and $\Delta \cdot f'_1(0)$. Thus, for the base side of the triangle we obtain

$$g = \Delta \cdot (|f'_{-1}(0)| + |f'_1(0)|) \tag{1}$$

If the height of the triangle and Equation 1 are substituted into the formula for the area of a triangle, the result is

$$\begin{aligned} loss &= \frac{1}{2} \cdot g \cdot h \\ &= \frac{1}{2} \cdot \Delta \cdot (\Delta \cdot (\left|f'_{-1}(0)\right| + \left|f'_1(0)\right|)) \\ &= \frac{1}{2} \cdot \Delta^2 \cdot (\left|f'_{-1}(0)\right| + \left|f'_1(0)\right|) \end{aligned} \tag{2}$$

Here, *loss* corresponds to the area of the triangle and thus to the net loss of classifier accuracy. Given that the error Δ is distributed with mean $M = 0$ and standard error $SE = \frac{SD}{\sqrt{n}}$, the mean of Δ^2 equals $(\frac{SD}{\sqrt{n}})^2$. This information can be inserted into Equation 2. Therefore, the expected net loss close to zero is approximated by

$$\begin{aligned} \mathbb{E}(loss) &= \frac{1}{2} \cdot (\frac{SD}{\sqrt{n}})^2 \cdot (|f'_{-1}(0)| + |f'_1(0)|) \\ &= \frac{SD^2(|f'_{-1}(0)| + |f'_1(0)|)}{2n} \end{aligned} \tag{3}$$

The error Δ is higher (that is, has a higher standard deviation and thus a higher expected absolute value) for smaller training data sets. A large error results in an also large net loss of accuracy. For larger training data sets, in contrast, the error Δ and thus the net loss of accuracy is minor. Therefore, it should be noted that the approximation of the expected net loss is more relevant for small numbers n of elements in the training set.

Equation 3 has the form of a hyperbolic function with respect to n, as generally described by

$$b + \frac{a}{n} \tag{4}$$

with $b = 0$ and $a = \frac{SD^2(|f'_{-1}(0)| + |f'_1(0)|)}{2}$. While the hyperbolic form itself is a very good approximation, note that the parameters a and b need to be estimated from the data.

Development of Alternative Options to Estimate Classifier Accuracy

The theoretically deduced knowledge that the classifier accuracy bias for small training data sets can be estimated by a hyperbolic function is used in this article to develop multiple options to estimate classifier accuracy. All these options have in common that the hyperbolic function

$$t(n) = b - \frac{a}{n}$$

is used as a model for the accuracy of the classifier (in contrast to Equation 4 above, which just modelled the accuracy loss) when trained with a training set of size n. As $n \to \infty$, the accuracy approaches an asymptote described by the parameter b. The parameter a indicates how fast the classifier approaches the asymptote; for a almost zero, the classifier is at maximal accuracy even for very few training samples, while larger a indicate that a high number of participants is needed to approach the asymptote. In the IV process, each tested data point can be used to estimate the parameter of the models. Ultimately, the asymptote parameter b is the one of interest; it describes how well the classifier works on the population, without bias from low initial sample sizes. If this value is larger than could be achieve by guessing, the two groups are different. In the following, multiple options to estimate the parameters when applying IV are introduced.

Least Square Estimation

Using the hyperbolic function, least square estimation can be used to estimate the parameters, including b depicting maximum classifier accuracy. The formal derivation is provided in the following paragraph.

Let $(i, r_i) \in \mathbb{N} \times 0, 1$ be data where i is the number of elements in the training set and r_i is 1 exactly if the classification is correct. We assume the model $b - \frac{a}{i}$ for the probability that r_i is one. The least squares distance is

$$L(a,b) = \sum_{i=1}^{n} \begin{cases} (1 - (b - \frac{a}{i}))^2 & r_i = 1 \\ (b - \frac{a}{i})^2 & r_i = 0 \end{cases} \tag{5}$$

with derivatives

$$\begin{aligned}\frac{\partial L(a,b)}{\partial a} &= \sum_{i=1}^{n} \frac{2}{i} \begin{cases} (1-(b-\frac{a}{i})) & r_i = 1 \\ -(b-\frac{a}{i}) & r_i = 0 \end{cases} \\ &= 2\sum_{i=1}^{n} (\frac{r_i}{i} - (\frac{b}{i} - \frac{a}{i^2})) \end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial L(a,b)}{\partial b} &= 2\sum_{i=1}^{n} -r_i + (b-\frac{a}{i}) \\ &= 2nb + 2\sum_{i=1}^{n} -r_i - \frac{a}{i} \end{aligned} \tag{7}$$

Setting the Equation 7 to zero yields

$$\begin{aligned} 0 &= nb - \sum_{i=1}^{n} r_i - a\sum_{i=1}^{n} \frac{1}{i} \\ b &= \frac{1}{n}\sum_{i=1}^{n} r_i + a\sum_{i=1}^{n} \frac{1}{ni} =: \mathrm{suc} + ac \end{aligned} \tag{8}$$

As r_i is 1 for correct classifications and 0 otherwise, the sum of all r_i divided by the number of elements equals the success rate of the classification. Hence, the first summand of the equation is referred to as suc. The second sum (without the leading a) is defined as c for the next steps. Inserting Equation 8 into Equation 6 and equating to zero is

$$\begin{aligned} 0 &= \sum_{i=1}^{n} \frac{1}{i}(r_i - \mathrm{suc} - ac + a\frac{1}{i}) \\ 0 &= \sum_{i=1}^{n} \frac{r_i - \mathrm{suc}}{i} + a\sum_{i=1}^{n} (-\frac{c}{i} + \frac{1}{i^2}) =: d + ae \end{aligned} \tag{9}$$

Defining the first sum as d and the second as e, the parameter a of the hyperbolic function can be computed as

$$a = -\frac{d}{e} \tag{10}$$

The values for c, d, and e can be computed from a classification result. These values can then in turn be used to compute the parameters a and b of the hyperbolic function, using Equations 10 and 8.

Maximum Likelihood Estimation

In order to use all available classification data while correcting for bias in small training sets, the likelihood function

$$L(Data|a,b) = (\prod_{n_{succ}} b - \frac{a}{n}) \cdot (\prod_{n_{fail}} 1 - b + \frac{a}{n}) \tag{11}$$

was set up. In this function, n_{succ} is the number of correct classifications, and n_{fail} is the number of incorrect classifications after one run of IV. Thus, the first term of Function 11 is the product of the probabilities of correct classifications and the second term is the product of the probabilities of incorrect classifications. The likelihood function is therefore the product of the probabilities of each tested element to be correctly or incorrectly classified under the given parameters a and b of the function. *Data* contains the information whether the tested elements were classified correctly or incorrectly, in the form of a vector consisting of ones and zeros. The order of binary values in this vector corresponds to the order of classification results output in IV for the increasing number of elements in the training data set. The function depends on two parameters only, which can easily be seen are not overspecified if the data contains both successful and failed trials.

Since the parameter b depicts the asymptote, i.e. the maximum classifier accuracy, estimating it via maximum likelihood allows to estimate the population classifier accuracy. It is further possible to extend this method into a Bayesian approach by adding a prior to the maximum likelihood estimation.

The following sections describe two simulations studies, investigating the theoretical derivations in this and the previous sections.

Simulation Study 1

To verify whether the equation $\mathbb{E}(loss)$ approximates the expected net loss, it was tested using a simulation. The goal was to compare the expected net losses based on $\mathbb{E}(loss)$ (see Equation 3) and the losses when simulating 100,000 runs for different numbers n of elements in the training data set.

Data Generation

The simulation was done using Java (Arnold et al., 2005). The simulation is split into two parts. At first, the underlying data is generated. This data is used for the second step, namely the simulation of the learning and testing phase of an classifier using IV. Univariate data was simulated from a normal distribution for both groups. Using two groups instead of more groups was done for three reasons, (1) it is the most basic case of classification, (2) the application onto studying group differences is straightforward, and (3) most of the characteristics of more general issues are already included in the two-group problem (Flach, 2017). Group sizes were fixed to be equal in the study to avoid bias in the results. In the following, the simulation process is explained step by step.

The maximum number of data samples $n_{max} = 40$ was chosen in the initial training data set. This cutoff choice was made based on the rule of thumb of $n \approx 20$ as proposed by Kim and von Oertzen (2018). In order to eliminate dependencies between the individual runs, the program drew a new random number for the feature x for each of the 100,000 runs. The mean values for the two normal distributions of the characteristics were set as $M_{-1} = -1$ for group -1 and $M_1 = 1$ for group 1. The standard deviation was set to $SD = 1$ for both groups.

Each simulation starts with $n = 2$ elements in the training dataset. In the first iteration, an element of the class $G = -1$ was created. For this element, univariate data was drawn from the underlying distribution for the characteristic x. Subsequently, two elements of the class $G = 1$ were produced, and random values were drawn for these two elements from the respective defined distribution for the feature x. Then, the element of the class $G = -1$ and the first element of the class $G = 1$ were used to train the classifier. The data point produced last, which always belonged to the class $G = 1$, was used for testing the classifier.

This process was repeated 100,000 times for each $n \in \{2, 4, 6, ..., n_{max}\}$, respectively. The number n is increased in steps of two in each run to ensure that there were always equal numbers of elements from both groups represented in the training data set. Each run produced an additional element of class $G = 1$ to be tested (see Appendix A for the script).

LDA was used as the classifier in the simulation. Since random data was drawn from normal distributions and the underlying normal distributions in both groups had the same standard deviations of $SD = 1$ for the variable, LDA is appropriate for this data (see also Boedeker & Kearns, 2019). For other data, different classifiers may give best results. In LDA, the mean values were used to classify the test data points. The datasets generated during the current study are available in the Supplementary Materials.

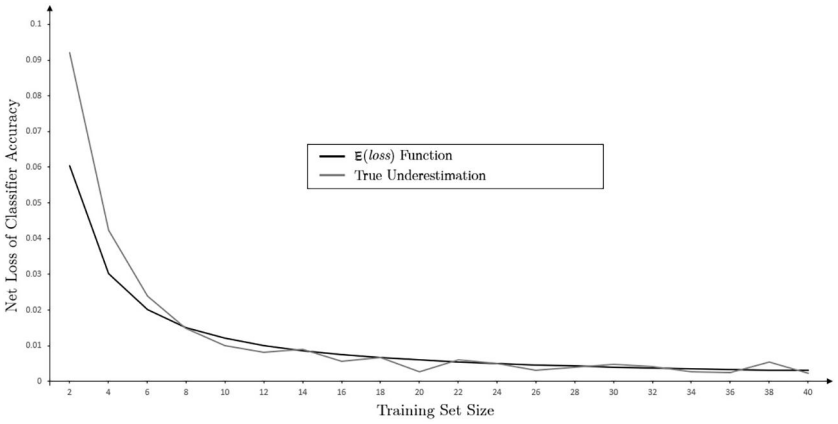
The expected net loss in the simulation was computed as the average distance of the best group separation resulting from the position of the two normal distributions and the actual average performance of the classifier in the simulation.

Results

As Figure 3 illustrates, the equation $\mathbb{E}(loss)$ approximates the expected net loss well for larger training sets. This is apparent from the minor differences between the two curves from as early as $n \approx 8$. However, for small training data set sizes ($n < 8$), the equation $\mathbb{E}(loss)$ underestimates the expected loss of accuracy. This is consistent with the theoretical consideration that the approximation is less accurate for very small training set sizes.

Figure 3

Approximation of Classifier Accuracy Bias



Note. For each training set size, 100,000 iterations were simulated.

Simulation Study 2

Further research focused on developing a method to more accurately estimate the optimal accuracy based on the correctness of the classification results, while using as much of the test data as possible. To this end, five different methods, which estimate the classifier accuracy, were compared, two existing and three newly developed ones. The existing methods included the computation of the overall accuracy, accumulated over the whole test set, and the approach suggested by Kim and von Oertzen (2018) to base the accuracy only on training data sets with $n \geq 20$. Both these options include a small bias since the initial training set size is small. To eliminate this bias, we added the three new options mentioned above. As the first new option, least squares estimation was used to compute the parameter values of the hyperbolic function. The second option was based on the developed maximum likelihood function in combination with an optimization algorithm.

Thirdly, the maximum likelihood function was extended by a $\beta(1.05, 1.05)$ prior, that is, a Beta distribution with parameters just above one. Adding this prior to the likelihood transforms the method into a Bayesian estimation of the asymptote parameter, which is the population accuracy of the classifier. The Bayesian approach hence allows to interpret the estimation result directly as a posterior on the difference between the two groups, as described by the ability of a classifier to separate them. The Beta prior is commonly used for such tasks since it is the conjugated prior of a probability parameter. The prior hyperparameters of 1.05 create a mostly flat prior on the interval $[0, 1]$ with a

slight maximum at 0.5 and boundary minima at 0 and 1, which helps to avoid numerical problems for low n when all trials are hits or fails, respectively.

Data Generation and Accuracy Estimation

The data generation was similar to Simulation Study 1, creating a univariate sample with maximum size of $n_{max} = 40$. The optimal accuracy to be reached by the classifier was set to 0.65, 0.80, or 0.95, respectively. This was achieved by changing the mean differences and, therefore, the overlap between the sample distributions. A total of 2,000 trials were conducted. The classification result in the shape of a vector consisting of zeros and ones was then exported into R (R Core Team, 2021). R was then used to estimate the accuracy by the five different methods (see Appendix B for the script).

For each of the five estimation methods we computed the average absolute difference from the true optimal classifier accuracy and the average bias from the true value. The bias was computed as the average difference from the true value.

For the estimation of the classifier accuracy based on the whole test set, the total number of correct classifications was divided by the total number of classification. For the estimation based on $n \geq 20$, the number of correct classifications with a training set size of at least 20 was divided by the total number of classifications. The least square estimation was computed using Equations 9 and 8.

For the maximum likelihood estimation an optimization algorithm in combination with Equation 11 was used. The lower bound for the parameter a was set to -2, the upper bound to 7. These values were chosen based on past experience and depict a reasonable range for the parameter. As b is equal to the asymptote of the hyperbolic function and this in turn corresponds to the estimated accuracy, b was limited to be larger than 0 and smaller than 1, since the accuracy cannot be worse than 0% or better than 100%. The results of the least squares estimation were used as starting values for the algorithm, unless the least squares estimate exceeded the chosen limits. In these cases, a value of -1.99999 or 6.99999 was chosen for a, and 0.00001 or 0.99999 for b, depending on whether the lower or upper limit had been exceeded. The optimization algorithm was provided with the $-log(L)$, to improve its technical functionality. The optimization algorithm returns optimal values for a and b. The value of b is an estimate of the classifier accuracy.

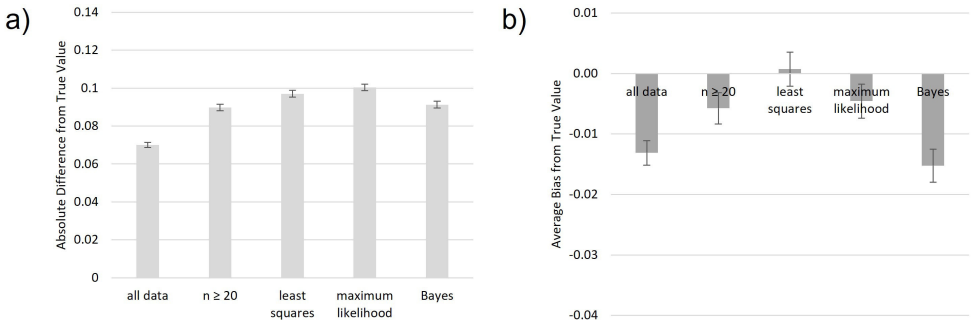
The Bayesian estimation of the classifier accuracy was conducted similarly to the maximum likelihood estimation. The only difference was that a $\beta(1.05, 1.05)$ prior was added to the computation of the likelihood. This prior was chosen, as it depicts that classifiers rarely if ever are able to reach a perfect estimation accuracy of 100%. The datasets generated during the current study are available in the Supplementary Materials.

Results

The results of the comparison are presented in Figure 4. Table 1 shows the average absolute differences between the estimation and the true classifier accuracy. A value of zero indicates a perfect estimation. The higher the average absolute difference is, the stronger does the estimation differ from the true classifier accuracy. Table 2 shows the average bias of the estimation. Negative values for the average bias indicate a systematic underestimation of the classifier accuracy, positive values indicate an overestimation. A value of zero indicates that the estimation under- and overestimates the true classifier accuracy to the same degree. Overall, the average absolute difference and average bias of all five estimation methods are small. Further, no estimation method can clearly be identified as the best option. While the estimation based on the whole test set depicts the smallest absolute difference, it also shows a stronger underestimation bias compared to the other methods. The least square estimation shows the smallest bias, but one of the largest average absolute differences.

Figure 4

Absolute Difference and Bias From the True Accuracy for Different Estimation Methods



Note. Based on a simulation of 2,000 trials with $N_{max} = 40$. Error bars depict standard errors.

In addition to the overall results, Figure 5 shows the absolute difference and average bias separately for a small (0.65), medium (0.80), and large (0.95) true classifier accuracy. The values are also depicted in Table 1 and Table 2. For small classifier accuracy using all data has the smallest absolute difference, but largest underestimation. Maximum likelihood and Bayes show the smallest bias, but a slightly higher absolute difference. For a high classifier accuracy using all data again shows the smallest absolute difference, here combined with one of the smaller biases. While Bayesian estimation shows one of the smaller absolute differences, but the largest underestimation.

Table 1

Mean and Standard Error of the Absolute Difference From the True Classifier Accuracy for Different Estimation Methods

	Total		Accuracy = .65		Accuracy = .80		Accuracy = .95	
Estimation method	***M***	***SE***	***M***	***SE***	***M***	***SE***	***M***	***SE***
Using all data	0.070	0.001	0.093	0.002	0.077	0.001	0.040	0.002
Using $n \geq 20$	0.090	0.002	0.116	0.002	0.095	0.002	0.058	0.002
Least squares	0.097	0.002	0.112	0.002	0.115	0.002	0.064	0.002
Maximum likelihood	0.100	0.002	0.121	0.002	0.110	0.002	0.070	0.001
Bayes	0.091	0.002	0.119	0.002	0.105	0.002	0.049	0.001

Note. Based on a simulation of 2,000 trials with $n_{max} = 40$. *M* = mean. *SE* = standard error.

Table 2

Mean and Standard Error of the Bias From the True Classifier Accuracy for Different Estimation Methods

	Total		Accuracy = .65		Accuracy = .80		Accuracy = .95	
Estimation method	***M***	***SE***	***M***	***SE***	***M***	***SE***	***M***	***SE***
Using all data	-0.013	0.002	-0.026	0.003	-0.011	0.002	-0.002	0.001
Using $n \geq 20$	-0.006	0.003	-0.010	0.003	-0.008	0.003	0.001	0.002
Least squares	0.001	0.003	-0.004	0.003	0.003	0.003	0.004	0.002
Maximum likelihood	-0.005	0.003	0.001	0.003	-0.004	0.003	-0.011	0.002
Bayes	-0.015	0.003	-0.001	0.003	-0.010	0.003	-0.035	0.002

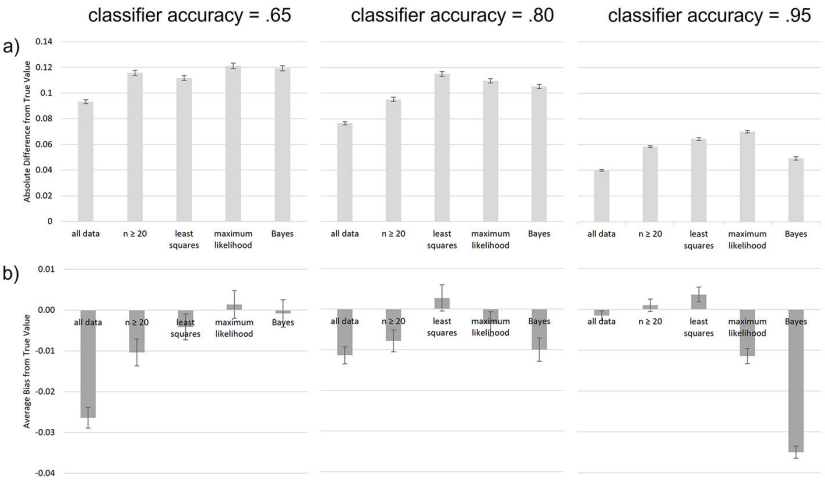
Note. Based on a simulation of 2,000 trials with $n_{max} = 40$. *M* = mean. *SE* = standard error.

Discussion

We investigated the estimation of classifier accuracy in the context of IV at small initial training set sizes to derive recommendations for dealing with the bias in these situations. A hyperbolic function describing the accuracy bias for small training data sets was derived mathematically and empirically confirmed through a simulation study. The hyperbolic function was then used to develop new methods to estimate the optimal classifier accuracy, accounting for an underestimation when test set sizes are small. Two existing estimation methods, computing overall accuracy and computing the accuracy for training set sizes with $n \geq 20$, were compared with a least square estimation, a maxi-

Figure 5

Absolute Difference and Bias From the True Accuracy for Different Estimation Methods and Different Accuracies



Note. Based on a simulation of 2,000 trials with $N_{max} = 40$. Error bars depict standard errors.

mum likelihood estimation, and a Bayesian estimation of the hyperbolic function. While overall the absolute differences and average biases were generally small, no estimation method could clearly be identified as the best one.

Independent Validation of Classifiers

IV has been suggested as an alternative to CV (Kim & von Oertzen, 2018), given that CV is subject to a bias based on the dependencies between training and test sets (Lanka et al., 2020). Nevertheless, IV and all other validation methods are also subject to a different source of bias, as it underestimates the classifier accuracy for small training set sizes. This occurs because of the insufficient training of the classifier due to the small training set. The present work presents an equation to estimate this underestimation, which has been shown to be reliable from a test set size of $n \geq 8$. As an underestimation of the classifier accuracy favors falsely rejecting the null hypothesis, the note by Kim and von Oertzen (2018) that the choice of the initial training set size is a critical decision for the outcome of IV must also be emphasized based on the results found in this study. Yet, the decision is less critical than initially thought.

Estimating Classifier Accuracy Using Independent Validation

We compared five different methods to estimate the classifier accuracy in the present study. Computing the overall accuracy using all of the available data yielded surprisingly good results, even though the classifier accuracy will be underestimated for small training set sizes. Excluding the data for training set sizes with $n < 20$ lead to a smaller bias in the estimation, yet the overall absolute difference from the true accuracy was larger. Using least squares estimation, maximum likelihood, or Bayesian estimation led to similar absolute differences as using $n \geq 20$ and maximum likelihood also showed a similar bias. Least square estimation could be labeled as an overall unbiased estimation, while the bias for Bayesian estimation was the largest. Yet, the bias was at maximum 1.5%.

The five estimation methods were differently well suited for low, medium, or high classifier accuracy. The maximum likelihood or Bayesian approach was best suited for a small classifier accuracy. Differences between estimation methods were smaller for medium classifier accuracy. For high classifier accuracy, the chosen Bayesian approach showed the overall largest bias with an underestimation of 3.5%. This is due to the specific prior we chose, a $\beta(1.05, 1.05)$ distribution. This prior is suitable to adjust for the increased risk of extreme classifier accuracies of 100% or 0%, which are more likely to occure falsely for small data sets. Yet, when the true accuracy is close to 100% it will lead to an underestimation. A different prior would likely yield better results in this case.

Overall, we would advise for the use of the least square estimation. It is overall an unbiased estimation and also shows good results when separately investigating small, medium, and high classifier accuracies. It has also the advantage of not requiring optimization processes, as is the case for the maximum likelihood and Bayesian approach. It is further based on all the available data. If the classifier accuracy is known to be low, maximum likelihood or Bayesian estimation is also a good choice. If the classifier accuracy is known to be high, simply computing the overall accuracy using all the available data also yields good results.

Limitations

Despite the previously described findings and the development of a new method for estimating classifier accuracy in the context of IV, some limitations exist with respect to the study.

Only two groups were used in this study. However, classifiers can also be used in situations where more than two classes are present. This aspect remains unsolved, although according to Flach (2017) most characteristics of more general problems are already included in binary problems.

Furthermore, all simulations, as well as the theoretical reasoning, were simple cases. Univariate normally distributed data was considered, the standard deviations of the distributions underlying the characteristics were chosen to be identical for both groups,

and the group sizes were defined to be equal. In contrast, when using real data, the distributions of the characteristics, as well as the group sizes, are usually different, and more than one variable is typically included in the data set. In the present study, simple cases were chosen to obtain a clearer view on the estimation of the accuracy using small training sets. In more complex conditions, the results would have additionally been affected by these factors.

Lastly, our approach assumes that the classifier uses a threshold to separate the groups. While this is true for LDA, the classifier we chose, it is not true for every classifier, as, for example, k-nearest-neighbors. In such cases our deduction can only serve as an analogy and it is unclear if the estimation would be reliable.

Conclusion

Classifiers, among other applications, can also be used to study group differences and are particularly promising in this area in some respects. Since previously existing validation methods were problematic, especially in the application of statistical tests, Kim and von Oertzen (2018) developed IV to circumvent the problems with this method. However, the accuracy determined in the validation process is subject to bias in small training sets, similar to all other validation techniques. Therefore, this study investigated the accuracy estimation on small training datasets.

Funding: The authors have no funding to report.

Acknowledgments: The authors have no additional (i.e., non-financial) support to report.

Competing Interests: Timo von Oertzen is Managing Editor-in-Chief of QCMB but played no editorial role for this particular article.

Preregistration: None of the experiments was preregistered.

Data Availability: The data files used for this article are freely available and can be found in the Supplementary Materials.

Supplementary Materials

For this article, the materials provided are the datasets for the two simulation studies, and the second independent validation simulation study in R (see Braun et al., 2022).

Index of Supplementary Materials

Braun, T., Eckert, H., & von Oertzen, T. (2022). *Independent Validation as a Validation Method for Classification* [Datasets, R simulation]. OSF. https://osf.io/ba7z4/

References

Arnold, K., Gosling, J., & Holmes, D. (2005). *The Java programming language*. Addison Wesley Professional.

Bax, E. (2012). Validation of k-nearest neighbor classifiers. *IEEE Transactions on Information Theory, 58*(5), 3225–3234. https://doi.org/10.1109/TIT.2011.2180887

Bay, S. D., & Pazzani, M. J. (2001). Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery, 5*(3), 213–246. https://doi.org/10.1023/A:1011429418057

Beins, B. C., & McCarthy, M. A. (2019). *Research methods and statistics in psychology* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/9781108399555

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Vol. 4). Springer.

Boedeker, P., & Kearns, N. T. (2019). Linear discriminant analysis for prediction of group membership: A user-friendly primer. *Advances in Methods and Practices in Psychological Science, 2*(3), 250–263. https://doi.org/10.1177/2515245919849378

Boucheron, S., Bousquet, O., & Lugosi, G. (2005). Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics, 9*, 323–375. https://doi.org/10.1051/ps:2005018

Bouckaert, R. R. (2003). Choosing between two learning algorithms based on calibrated tests. In Fawcett, T. & Mishra, N. (Eds.), *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 51–58). AAAI Press.

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly, 36*(4), 1165–1188. https://doi.org/10.2307/41703503.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018

Counsell, A., & Harlow, L. L. (2017). Reporting practices and use of quantitative methods in Canadian journal articles in psychology. *Canadian Psychology/Psychologie canadienne, 58*(2), 140–147. https://doi.org/10.1037/cap0000074

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*(2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Flach, P. (2017). *Machine learning: The art and science of algorithms that make sense of data* (11th ed.). Cambridge University Press.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition* (pp. 278–282). IEEE. https://doi.org/10.1109/ICDAR.1995.598994

Hoekstra, R., Kiers, H. A. L., & Johnson, A. (2012). Are assumptions of well-known statistical techniques checked, and why (not)? *Frontiers in Psychology, 3*, Article 137. https://doi.org/10.3389/fpsyg.2012.00137

Huberty, C. J., & Hussein, M. H. (2003). Some problems in reporting use of discriminant analyses. *Journal of Experimental Education, 71*(2), 177–192. https://doi.org/10.1080/00220970309602062

Kim, B., & von Oertzen, T. (2018). Classifiers as a model-free group comparison test. *Behavior Research Methods, 50*(1), 416–426. https://doi.org/10.3758/s13428-017-0880-z

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *IJCAI'95: Proceedings of the 14th International Joint Conference on Artificial intelligence* (Vol. 2, pp. 1137–1143). Morgan Kaufmann. https://dl.acm.org/doi/10.5555/1643031.1643047

Lanka, P., Rangaprakash, D., Dretsch, M. N., Katz, J. S., Denney, T. S., Jr., & Deshpande, G. (2020). Supervised machine learning for diagnostic classification from large-scale neuroimaging datasets. *Brain Imaging and Behavior, 14*(6), 2378–2416. https://doi.org/10.1007/s11682-019-00191-8

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106. https://doi.org/10.1007/BF00116251

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.. https://www.R-project.org/

Sahiner, B., Chan, H.-P., & Hadjiiski, L. (2008). Classifier performance estimation under the constraint of a finite sample size: Resampling schemes applied to neural network classifiers. *Neural Networks, 21*(2-3), 476–483. https://doi.org/10.1016/j.neunet.2007.12.012

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery, 1*(3), 317–328. https://doi.org/10.1023/A:1009752403260

Santafe, G., Inza, I., & Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review, 44*(4), 467–508. https://doi.org/10.1007/s10462-015-9433-y

Wee, A. G. (2000). Comparison of impression materials for direct multi-implant impressions. *Journal of Prosthetic Dentistry, 83*(3), 323–331. https://doi.org/10.1016/S0022-3913(00)70136-3

Weisberg, Y. J., DeYoung, C. G., & Hirsh, J. B. (2011). Gender differences in personality across the ten aspects of the Big Five. *Frontiers in Psychology, 2*, Article 178. https://doi.org/10.3389/fpsyg.2011.00178

Zhao, M., Wang, M., Zhang, J., Gu, J., Zhang, P., Xu, Y., Ye, J., Wang, Z., Ye, D., Pan, W., Shen, B., He, H., Liu, M., Liu, M., Luo, Z., Li, D., Liu, J., & Wan, J. (2020). Comparison of clinical characteristics and outcomes of patients with coronavirus disease 2019 at different ages. *Aging, 12*(11), 10070–10086. https://doi.org/10.18632/aging.103298

Appendices

Appendix A: Java Script

The following Java script was used for Simulation Study 1 and to generate the classifier outcome for Simulation Study 2.

```
package classifier;

import java.util.Random;
import org.apache.commons.math3.distribution.NormalDistribution;
import java.io.IOException;
import java.io.FileWriter;

public class SimComparisonOfAccEstMethCorrected {

    static Random zufallx = new Random();

    public static void main(String[] args) throws IOException {
            getData();
    }

    public static void getData() throws IOException {

            for (int z = 1; z <=2000; z += 3) {
                int counter = 10000+z;
                FileWriter myWriter = new FileWriter ("simulation_"+counter+".txt");
                myWriter.write("n"+"␣"+"correct"+"␣"+"true.value");
                myWriter.write("\n");
                int [] sum = getClassificationResultsReal246(0.770640933);
                int [] N = getN(40);
                double accopt = getAccOpt (0.770640933);
                for (int i = 0; i<20; i++) {
                        myWriter.write(N[i]+"␣"+sum[i]+"␣"+accopt);
                        myWriter.write("\n");
                        if (i==19) myWriter.close();
                }
            }

            for (int z = 2; z <=2000; z += 3) {
                int counter = 10000+z;
                FileWriter myWriter = new FileWriter ("simulation_"+counter+".txt");
                myWriter.write("n"+"␣"+"correct"+"␣"+"true.value");
                myWriter.write("\n");
                int [] sum = getClassificationResultsReal246(1.683242467);
                int [] N = getN(40);
                double accopt = getAccOpt (1.683242467);
                for (int i = 1; i<20; i++) {
                        myWriter.write(N[i]+"␣"+sum[i]+"␣"+accopt);
                        myWriter.write("\n");
                        if (i==19) myWriter.close();
                }
            }

            for (int z = 3; z <=2000; z += 3) {
                int counter = 10000+z;
                FileWriter myWriter = new FileWriter ("simulation_"+counter+".txt");
                myWriter.write("n"+"␣"+"correct"+"␣"+"true.value");
                myWriter.write("\n");
                int [] sum = getClassificationResultsReal246(3.289707254);
                int [] N = getN(40);
                double accopt = getAccOpt (3.289707254);
                for (int i = 1; i<20; i++) {
```

```
                    myWriter.write(N[i]+"␣"+sum[i]+"␣"+accopt);
                    myWriter.write("\n");
                    if (i==19) myWriter.close();
                }
            }
            System.out.println("Done");
    }

    public static double getAccOpt(double mean) {
            double stdv11 = 1.0, mean11 = 0;
            double mu = mean;
            double accOpt = calcAccOpt (mu, mean11);
            return accOpt;
    }

    public static int [] getN(int its) {
            int [] N = new int [its/2];
            for (int n = 0; n<its/2; n++) {
               N[n] = (n+1)*2;
            }
            return N;
    }

    // Classifies all data in steps of two
    public static int [] getClassificationResultsReal246 (double mean) {
            double stdv11 = 1.0, mean11 = 0;
            double mu = mean;
            double accOpt = calcAccOpt (mu, mean11);
            double stdv21 = 1.0, mean21 = mu;
            int s = 40;
            int [] correctClassification = new int [s/2];
            int[] group = new int[42];
               double[][] feature = new double[42][1];

            for (int i=0, a=1; i<41 && a<=41; i=i+2, a=a+2) {
               group[i]=-1;
               group[a]=1;
               feature[i][0]=randomXample(stdv11, mean11);
               feature[a][0]=randomXample(stdv21, mean21);
            }

            for (int i = 2; i<=41; i+=2) {
               LinearDiscriminantAnalysis lda = new
                  LinearDiscriminantAnalysis(feature, group);
               lda.ignoreAPrioris = true;
               lda.ignoreCovariance = true;
               lda.useLinearClassifier = true;
               lda.trainData(0, i);
               int r = lda.classify(feature[i]);
               boolean correct = (r == group[i]);
               correctClassification[i/2-1] = correct ? 1:0;
               if (i == 40) System.out.println(lda.leaveOneOutBAC());
            }
            return correctClassification;
    }
```

```
    // Classifies all data, beginning at the second
    public static int [] getClassificationResultsReal234 (double mean) {
            double stdv11 = 1.0, mean11 = 0;
            double mu = mean;
            double accOpt = calcAccOpt (mu, mean11);
            double accOpt = calcAccOpt (mu, mean11);
            double stdv21 = 1.0, mean21 = mu;
            int s = 40;
            int [] correctClassification = new int [s];
            int[] group = new int[41];
               double[][] feature = new double[41][1];

            for (int i=0, a=1; i<=40 && a<=40; i=i+2, a=a+2) {
               group[i]=-1;
               group[a]=1;
               feature[i][0]=randomXample(stdv11, mean11);
               feature[a][0]=randomXample(stdv21, mean21);
            }

            for (int i = 2; i<=40; i++) {
               LinearDiscriminantAnalysis lda = new
                  LinearDiscriminantAnalysis(feature, group);
               lda.ignoreAPrioris = true;
               lda.ignoreCovariance = true;
               lda.useLinearClassifier = true;
               lda.trainData(0, i);
               int r = lda.classify(feature[i]);
               boolean correct = (r == group[i]);
               correctClassification[i-2] = correct ? 1:0;
               if (i == 40) System.out.println(lda.leaveOneOutBAC());
            }
            return correctClassification;
    }

    public static double randomXample(double stdv1, double mean1) {
            double zufallszahlx = zufallx.nextGaussian() * stdv1 + mean1;
            return zufallszahlx;
    }

    public static double calcAccOpt (double mu, double mean11) {
            NormalDistribution nv = new NormalDistribution(mean11,1.0);
            double accOpt = nv.cumulativeProbability(mu/2);
            return accOpt;
    }

    public static double drawRandomUniformValue (double min, double max) {
            double random = new Random().nextDouble();
            double u = min + (random * (max - min));
            double uscaled = u * 1;
            double urounded = Math.round(uscaled * 10000.0)/10000.0;
            return urounded;
    }

}
```

Appendix B: R Script

The following R script was used for Simulation Study 2.

```
iter <- 2000
results <- data.frame(matrix(0, iter, 9))
colnames(results) <- c("ML.b", "ML.a", "MLprior.b", "MLprior.a",
                       "Larger20.b", "All.b", "LS.b", "LS.a", "Original.b")

# Maximum Likelihood Estimation
Likeli <- function(par, data) {
     b=par[1]
     a=par[2]

     if(b > 1 | a>7 | a < -2) {
            return(100)
     } else{
            p<-1
            for (i in 1:length(data.class$n)) {
               q <- b-a/data$n[i]
               q <- ifelse(q>0, q, 0.00001)
               q <- ifelse(q>1, 0.99999, q)
               p <- ifelse(data$correct[i]>0,
               p*q,
               p*(1-q))
            }
            return(-log(p))
     }
}

# Bayesian estimation
Likeli.prior <- function(par, data){
   b=par[1]
   a=par[2]

   if(b < 0 | b > 1 | a>7 | a < -2){
            return(100)
   } else{
            p<-1
            for (i in 1:length(data.class$n)) {
               q <- b-a/data$n[i]
               q <- ifelse(q>0, q, 0.00001)
               q <- ifelse(q>1, 0.99999, q)
               p <- ifelse(data$correct[i]>0,
               p*q,
               p*(1-q))
            }
            p <- p*dbeta(b, 1.05, 1.05, ncp = 0)
            return(-log(p))
   }
}

# Simulation
for(i in 1:iter) {
   nr <- 10000 + i
```

```
    data.class <- read.table(paste0("simulation_",nr,".txt"), header = T)
    sum.correct.larger20 <- 0
    n.larger20 <- 0
    for(j in 1:length(data.class$n)) {
            if(data.class$n[j] >= 20){
                   sum.correct.larger20 <- sum.correct.larger20
                               + data.class$correct[j]
                   n.larger20 <- n.larger20 + 1
            }
    }
    results[i,5] <- sum.correct.larger20/n.larger20
    results[i,6] <- sum(data.class$correct)/length(data.class$correct)
    # Least Squares etimation
    data.class$inv.n <- 1/data.class$n
    c <- sum(data.class$inv.n)/length(data.class$inv.n)
    data.class$step1 <- (data.class$correct-results[i,6])/data.class$n
    d <- sum(data.class$step1)
    data.class$step2 <- 1/(data.class$n*data.class$n)-c/data.class$n
    e <- sum(data.class$step2)
    results[i,8] <- -d/e
    results[i,7] <- results[i,6]+results[i,8]*c
    # Maximum Likelihood Estimation
    b.start <- ifelse(results[i,7]>=1,0.99999, results[i,7])
    a.start <- ifelse(results[i,8]<-2, -1.99999,
                          ifelse(results[i,8]>7,6.99999,results[i,8]))
    resultLikeli <- optim(par = c(b.start, a.start), Likeli,
                                              data = data.class)
    results[i,1] <- resultLikeli$par[1]
    results[i,2] <- resultLikeli$par[2]
    # Bayesian estimation
    resultLikeli.prior <- optim(par = c(b.start, a.start), Likeli.prior,
                             data = data.class)
    results[i,3] <- resultLikeli.prior$par[1]
    results[i,4] <- resultLikeli.prior$par[2]
    # True value
    results[i,9] <- data.class$true.value[1]
}

write.table(results, "ResultsRaw.txt")

# Absolute Difference
abs.diff.results <- data.frame(matrix(0, iter, 6))
colnames(abs.diff.results) <- c("Diff.ML.b", "Diff.MLprior.b",
        "Diff.Larger20.b", "Diff.All.b", "Diff.LS.b", "Original.b")

abs.diff.results$Diff.ML.b <- abs(results$ML.b - results$Original.b)
abs.diff.results$Diff.MLprior.b <- abs(results$MLprior.b - results$Original.b)
abs.diff.results$Diff.Larger20.b <- abs(results$Larger20.b
                                        - results$Original.b)
abs.diff.results$Diff.All.b <- abs(results$All.b - results$Original.b)
abs.diff.results$Diff.LS.b <- abs(results$LS.b - results$Original.b)
abs.diff.results$Original.b <- results$Original.b

write.table(abs.diff.results, "ResultsAbsDifference.txt")

# Bias
```

```
diff.results <- data.frame(matrix(0, iter, 6))
colnames(diff.results) <- c("Diff.ML.b", "Diff.MLprior.b",
        "Diff.Larger20.b", "Diff.All.b", "Diff.LS.b", "Original.b")

diff.results$Diff.ML.b <- (results$ML.b - results$Original.b)
diff.results$Diff.MLprior.b <- (results$MLprior.b - results$Original.b)
diff.results$Diff.Larger20.b <- (results$Larger20.b - results$Original.b)
diff.results$Diff.All.b <- (results$All.b - results$Original.b)
diff.results$Diff.LS.b <- (results$LS.b - results$Original.b)
diff.results$Original.b <- results$Original.b

write.table(diff.results, "ResultsDifference.txt")
```