Measuring the Stability of Results from Supervised Statistical Learning

Philipp, Michel; Rusch, Thomas; Hornik, Kurt; Strobl, Carolin

Published: 01/01/2017

Document Version
Publisher's PDF, also known as Version of record

Measuring the Stability of Results from Supervised Statistical Learning

Michel Philipp, Thomas Rusch, Kurt Hornik, Carolin Strobl

Research Report Series
Report 131, January 2017

Institute for Statistics and Mathematics
http://statmath.wu.ac.at/

Measuring the Stability of Results from Supervised Statistical Learning

Michel Philipp
Department of Psychological Methods, Evaluation, and Statistics
University of Zurich
and
Thomas Rusch
Competence Center for Empirical Research Methods
WU Vienna University of Economics and Business
and
Kurt Hornik
Institute for Statistics and Mathematics
WU Vienna University of Economics and Business
and
Carolin Strobl
Department of Psychological Methods, Evaluation, and Statistics
University of Zurich

January 17, 2017

Abstract

Stability is a major requirement to draw reliable conclusions when interpreting results from supervised statistical learning. In this paper, we present a general framework for assessing and comparing the stability of results, that can be used in real-world statistical learning applications or in benchmark studies. We use the framework to show that stability is a property of both the algorithm and the data-generating process. In particular, we demonstrate that unstable algorithms (such as recursive partitioning) can produce stable results when the functional form of the relationship between the predictors and the response matches the algorithm. Typical uses of the framework in practice would be to compare the stability of results generated by different candidate algorithms for a data set at hand or to assess the stability of algorithms in a benchmark study. Code to perform the stability analyses is provided in the form of an R-package.

Keywords: Resampling, Recursive Partitioning, R-package **stablelearner**

1 Introduction

Influential statisticians have previously pointed out the importance of stability to draw reliable conclusions (or more generally speaking for reproducibility) when interpreting results from statistical learning [see, e.g., Stodden, 2015, Turney, 1995, Yu, 2013]. Yu [2013], for example, stated: "More often than not, modern scientific findings rely on statistical analysis of high-dimensional data, and reproducibility is imperative for any scientific discovery. Scientific reproducibility therefore is a responsibility of statisticians." (p. 1485). To meet this demand in practical applications as well as in methodological research, we here present a framework that can be used for measuring the stability of results from statistical learning methods. In simple terms, stability is revealed when the interpretation of results generated by using different data sets drawn from some data-generating process (DGP) lead to identical or at least very similar conclusions.

Algorithmic methods (e.g., recursive partitioning, support vector machines, neural networks, and k-nearest neighbors) have become widely used [Kuhn and Johnson, 2013], either for *predictive modeling* or *explanatory modeling* [Shmueli, 2010]. In the former the algorithm is trained on a learning sample and then used for prediction on new observations, in the latter the goal is to learn about the nature of the DGP. We refer to Breiman [2001] and Shmueli [2010] for a discussion. In this paper we focus on the stability of results when conducting explanatory data analysis.

When a single result is interpreted in explanatory modeling, the aim is to draw conclusions about the nature of the DGP. For this to work well, stability is a major requirement, since we would expect to draw the same or at least very similar conclusions from interpreting results generated with a given algorithm on different random samples from the same DGP. Unfortunately, some algorithms are prone to generate unstable results [Breiman, 1996], i.e., small random changes in the data can cause large changes in the interpretation of a result. It is thus important to be able to gauge the reproducibility of the results in terms of stability.

Take one popular method, recursive partitioning [see, e.g., Strobl et al., 2009, for an introduction]. Here, the predictor space is repeatedly split to identify groups of observations with similar values in the response variable. The simplest cases are the well-known classification and regression trees [Breiman et al., 1984] but splitting can also be conducted with respect to the different parameter values in a particular model, for example, a generalized

linear model [e.g., Rusch and Zeileis, 2013] as done in model-based recursive partitioning [Hothorn et al., 2006, Zeileis et al., 2008]. The partition is usually illustrated in the form of a decision tree, that is used to make inferences about the DGP, but the results can be notoriously unstable and the inference questionable. Being able to assess and quantify the stability of a result is therefore critical.

In this paper, we present a general framework to investigate and quantify the stability of results, both when a data set (in real world supervised learning problems) or the DGP (in simulation studies) is available. The framework can support practitioners in judging the stability of a single analysis result or in choosing the most stable algorithm among a set of candidates with respect to a consistent interpretation of the results. In addition to this, it can also be used as a performance criterion in benchmark experiments [see Hothorn et al., 2005]. Code to perform the stability assessment in practice is provided in the form of a software package called **stablelearner** (currently available from `http://r-forge.r-project.org/projects/stablelearner/`) for the free open source software R for statistical computing [R Core Team, 2016].

The remainder of the article is organized as follows. We first present our theoretical considerations and our new framework for measuring stability in Section 2, and describe a variety of specifications of the framework in Section 3. Section 4 contains simulation experiments to demonstrate some important properties of the framework and an illustrative example with real data sets to show how the framework can be used in practice. Section 5 concludes the article with a discussion and ideas for future research.

1.1 Related work and contribution

The stability of algorithms for statistical learning has been studied in statistics and machine learning for quite some time [see, e.g., Bousquet and Elisseeff, 2002, Breiman, 1996, Mukherjee et al., 2006, Poggio et al., 2004] mainly by labeling certain algorithms as being stable or not. Ideas for measuring the stability of results have been presented previously in Turney [1995], Lange et al. [2002], Lim and Yu [2016], Briand et al. [2009] and Ntoutsi et al. [2008] either for the classification or the regression case, but so far not for both.

Our contribution extends this work along two lines: First, instead of focusing only on the algorithm, we view stability of a result to depend on both the algorithm and/or model and how well the algorithm matches the DGP. Stability therefore has several components:

algorithm, specified model, and DGP. We focus on empirically measuring the stability of a result from a practitioner's perspective. Second, we introduce a very general framework for assessing the stability of results from statistical learning that is applicable to most supervised learning situations in a similar manner. The procedure involves the pairwise comparison of results generated from learning samples randomly drawn from the original data set or – in the case of a simulation study – directly from the DGP. For the comparison of the results, a similarity (or dissimilarity) measure plus an additional evaluation sample will be required.

2 Stability measuring framework

Throughout this article, we focus on supervised learning problems by assuming predictor-response data $Z = (Y, X)$. Let Y denote the (possibly multivariate) response from sample space $\mathcal{Y}$ and $X = (X_1, \ldots, X_p)$ the p-dimensional vector of predictors from sample space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$. For generality, let us assume there exists a DGP based on the joint probability $\mathsf{P}_Z = \mathsf{P}_{Y,X}$ that characterizes the population of interest.

We further assume that the conditional distribution of the response $\mathsf{P}_{Y|X}$ depends on a function f of the predictors. The goal in explanatory modeling is to reliably estimate f, such that we can draw conclusions about the relation between X and Y. f represents the unknown function that describes the relationship between the predictors and an element of the conditional distribution, often the expectation.

Let $\mathcal{A}$ be an algorithm for statistical learning. Popular ones include algorithms from recursive partitioning, neural networks, support vector machines, k-nearest neighbors or various types of regression models.

Additionally, let $\mathcal{M}$ be the model that specifies the assumed relationship between the predictors and the response. At a minimum, $\mathcal{M}$ specifies the response and the predictor variables but may be more concrete for certain algorithms.

The goal in supervised learning is to approximate f by means of an algorithm $\mathcal{A}$ for the specified model $\mathcal{M}$ and a given learning sample $\mathfrak{L}' = \{z_1, \ldots, z_n\}$ generated by the DGP (where n is the number of observations). The result is a function denoted by $r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}') = \hat{f}(x)$, that is an estimate of the unknown function $f(x)$ and can be used to predict the response for new instances of x.

Now, suppose a different learning sample, say $\mathfrak{L}''$, was generated by the DGP. The result

from learning the algorithm $\mathcal{A}$ for the model $\mathcal{M}$ on the learning sample $\mathfrak{L}''$ is denoted by $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}'')$. Due to the sampling variability or to design principles of the algorithm (e.g., injection of randomness), the two results $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}')$ and $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}'')$ may be not the same. Both estimates of $f(x)$ may be different and the predictions may vary (at least for some x). Hence, one can think of $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}')$ and $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}'')$ as realizations of a random function $r_{\mathcal{A},\mathcal{M}}(x)$ in the domain of functions that could result from learning on different samples generated by the DGP. The distribution of $r_{\mathcal{A},\mathcal{M}}(x)$ depends on several components; most importantly on the algorithm, the specified model, the DGP, and the sample size n:

$$r_{\mathcal{A},\mathcal{M}}(x) \sim \mathsf{P}_r(\mathcal{A}, \mathcal{M}, \text{DGP}, n, \ldots).$$

In explanatory modeling, the result $r_{\mathcal{A},\mathcal{M}}(x;\mathfrak{L}')$ is interpreted to learn about the DGP. Then, the question arises whether the interpretation of the result would have been similar or equal if a different learning sample $\mathfrak{L}''$ had been used to obtain it.

The basic idea discussed in this article is to assess the stability of a result by quantifying the similarity of realizations from the distribution P_r, which requires to assess the similarity between the results generated by training the algorithm $\mathcal{A}$ on two different learning samples, $\mathfrak{L}'$ and $\mathfrak{L}''$, by means of a similarity measure. Large similarity indicates that two results lead to the same or similar conclusions about the underlying DGP and, thus, implies high stability. Before we present the procedure to assess the stability of a result, we discuss how two results can be compared with respect to their interpretation.

2.1 Semantic versus structural similarity

Semantic measures compare the logical meaning of two results by their prediction, structural measures compare the appearance of two results by their structural elements (e.g., by the nodes and the branches in a tree). The literature provides a vast number of semantic and structural measures to quantify the similarity between results [see, e.g., Banerjee et al., 2012, Briand et al., 2009, Miglio and Soffritti, 2004, Ntoutsi et al., 2008, Shannon and Banks, 1999, Turney, 1995, for a selection of approaches for recursive partitioning]. For our purpose, we recommend that a semantic measure should be used.

This recommendation is based on considerations similar to Turney [1995]: Two results that are structurally different can be logically equivalent and lead to the same interpretation. To illustrate this, we consider trees applied to a simple classification problem with

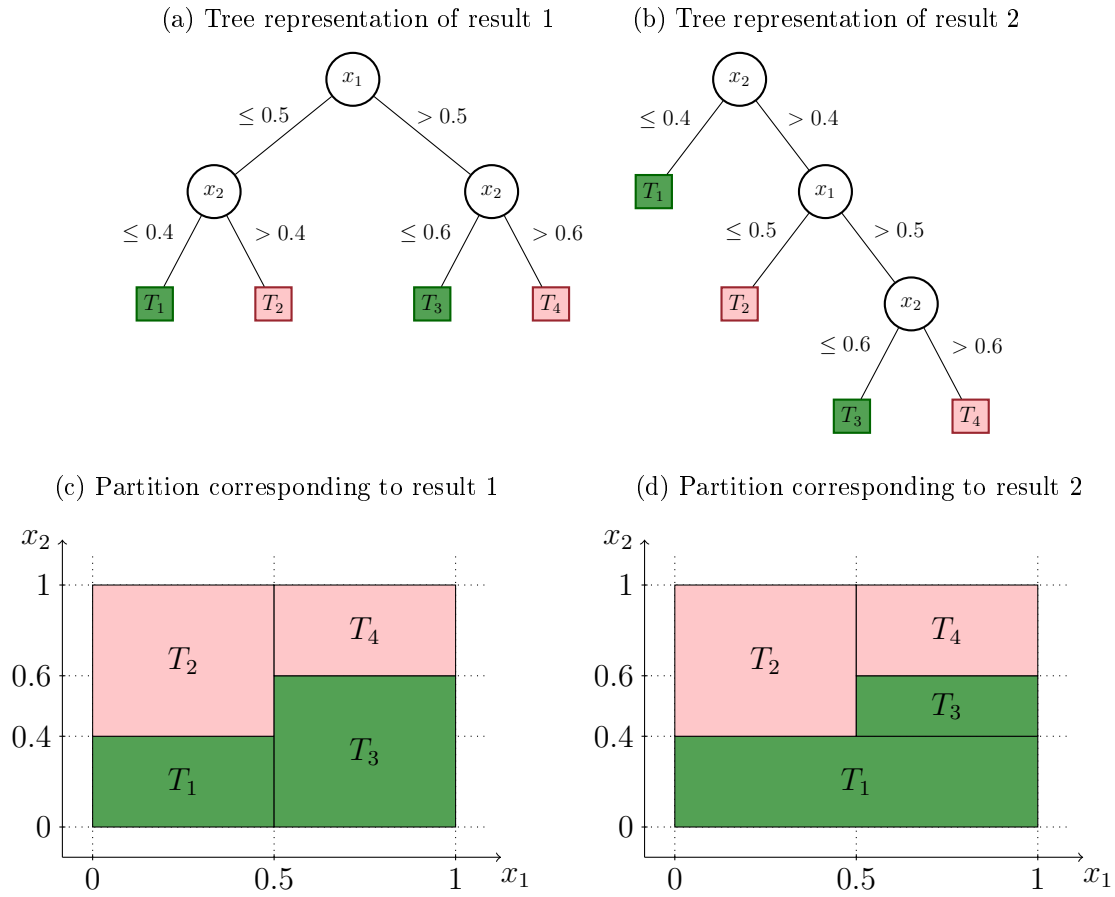

Figure 1: Examples of different tree structures, but equivalent partitions and interpretations.

two classes and a two-dimensional predictor space. Figure 1 shows the two exemplary trees (first row) and the corresponding partitions of the predictor space (second row), where the predicted class in each terminal node is indicated by two different colors.

As is apparent in the first row, the structures of the trees differ. The second row, however, shows that the predictions are equivalent for any point in the predictor space. Thus, the two trees have the exact same semantic and lead to the same substantive conclusions.

In addition to the argument above, fair comparisons of the stability of results between different types of algorithms are only feasible when a semantic similarity measure is used.

2.2 Measuring similarity based on predictions

As illustrated above, the similarity of two results is best assessed by comparing their predictions over the entire predictor space, or, by means of a semantic similarity measure [Turney, 1995]. A similarity measure is a function $s(\cdot, \cdot)$ that generates values on (a portion of) the real line. Measures that are particularly useful for computing the similarity of predictions

in the context of regression and classification problems are discussed in Section 3.1.

To compute the similarity between the predictions from the two results, an evaluation sample is needed. In choosing the evaluation sample one can draw from P_X, which puts appropriate emphasis one areas in the predictor space where new observations are more likely to occur, or from a particular region of the predictor space $\mathcal{X}$ of particular interest.

Let $\mathfrak{E}$ be such an evaluation sample that contains m observations, that is $|\mathfrak{E}| = m$. Then the predictions of the results are given by

$$\hat{y}' = \{r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}') : x \in \mathfrak{E}\} \quad \text{and} \quad \hat{y}'' = \{r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}'') : x \in \mathfrak{E}\},$$

and their similarity can be calculated by $s\big(\hat{y}', \hat{y}''\big)$.

As mentioned above, both results are realizations of a random function. The similarity between the predictions of the two results can therefore again be seen as a realization of a random variable S. The domain of possible similarity values depends on the similarity function and via the prediction of the results on the algorithm, the specified model, the DGP, and the sample size. Thus, the distribution of the similarity is given by

$$S \sim \mathsf{P}_S(s(\cdot,\cdot), \mathcal{A}, \mathcal{M}, \text{DGP}, n, \ldots).$$

Finally, the stability of a result for a given algorithm, specified model and DGP can be assessed by studying P_S or its characteristics. P_S may be unknown and needs to be approximated by generating many realizations $s(\hat{y}'_b, \hat{y}''_b)$ $(b = 1, \ldots, B)$, e.g., with the generic procedure described in Section 2.3.

2.3 Stability measurement procedure

Similar to Hothorn et al. [2005], we distinguish between the following two situations:

- The *real data problem* (as encountered in real-world statistical learning applications or in benchmark experiments with real data sets) where the DGP is unknown and all information available is a fixed set of n observations $\{z_1, \ldots, z_n\} \sim \mathsf{P}_{Z_n}$, that is the original data set.

- The *simulation study problem* (as encountered in benchmark experiments with artificial data sets) where the DGP is known precisely, such that an arbitrary number of learning samples $\{z_1, \ldots, z_n\} \sim \mathsf{P}_{Z_n}$ can be generated by the DGP.

For empirically measuring stability, we suggest the following generic procedure. For iteration $b = 1, \ldots, B$,

(1) Generate two learning samples $\mathfrak{L}_b'$ and $\mathfrak{L}_b''$ plus an evaluation sample $\mathfrak{E}_b$ by sampling from $\mathcal{F}$ that is a proxy for P_Z. To do this in practice, the plug-in principle can be used. Thus, for the

$$\begin{aligned} \text{real data problem:} \quad \mathcal{F} &\mathrel{\widehat{=}} \widehat{F}_n, \\ \text{simulation study problem:} \quad \mathcal{F} &\mathrel{\widehat{=}} \mathsf{P}_Z, \end{aligned}$$

where $\widehat{F}_n$ is an approximation of P_Z that represents the DGP increasingly better for increasing n.

(2) Generate the results $r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}_b')$ and $r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}_b'')$ by training the algorithm on both learning samples.

(3) Compute $s(\hat{y}_b', \hat{y}_b'')$ using the evaluation sample $\mathfrak{E}_b$.

Drawing samples from $\widehat{F}_n$ is equivalent to resampling from the original data set [see, e.g., Wasserman, 2004, chap. 8]. Thus, resampling from the original data set can be used to generate the learning and evaluation samples in the real data problem. Following the notation above, this is comparable to sampling directly from the DGP, as carried out in the simulation study problem.

To apply the procedure in practice, users have to choose a similarity measure $s(\cdot, \cdot)$ and, for real data problems, a resampling and an evaluation method. Sensible choices are presented in the following section.

3 Framework settings

In the first part of the following section we discuss a few exemplary options of similarity and dissimilarity measures for regression and classification problems. In the second part of the section, different methods for generating the learning and the evaluation samples for the case of a real data problem are presented.

3.1 Similarity and dissimilarity measures

We distinguish between similarity and dissimilarity measures, denoted by $s(\cdot, \cdot)$ and $d(\cdot, \cdot)$, respectively. Most similarity and dissimilarity measures have a lower and/or an upper

bound. Some measures are additionally normalized, for example between -1 and 1 (most correlation measures) or between 0 and 1 (some distance measures). Normalized measures have the additional advantage that their range has an absolute meaning.

For reasons of comparability, dissimilarity measures may need to be converted into similarity measures. Nonnormalized dissimilarity measures may be converted into similarity using $s(\cdot, \cdot) = -d(\cdot, \cdot)$. To convert a normalized dissimilarity measure into a normalized similarity, it may be appropriate to use the upper bound as a reference, that is, $s(\cdot, \cdot) = d_{max} - d(\cdot, \cdot)$, where d_{max} is the upper bound of the dissimilarity measure.

A practically relevant task is the comparison of stability assessments from different results. To compare the stability of two or more results generated using data sets or DGPs in which the responses were measured on different scales, we recommend to use a similarity measure that is invariant to changes in the scale. We denote a similarity measure as scale-invariant if $s(a + b \cdot \hat{y}', a + b \cdot \hat{y}'') = s(\hat{y}', \hat{y}'')$ for $a, b \neq 0$.

Please be aware that only *point-wise* similarity measures should be used for measuring stability in our framework, which guarantees that comparisons are made between predictions for the same point in the predictor space (as we argued in Section 2.1).

3.1.1 Regression

Let us first consider the regression case. Thus, in the univariate case, $\hat{y}'$ and $\hat{y}''$ are numeric vectors of length m ($\hat{y}', \hat{y}'' \in \mathbb{R}$). The literature provides a vast number of similarity and dissimilarity measures used in diverse fields (e.g., in medicine, psychology, image registration, clustering, etc.) to compare numeric measurements.

The well-known *Euclidean distance* (ED) is a dissimilarity measure that can be used to compute the similarity of numeric predictions. It is computed as

$$d_{\text{ED}}(\hat{y}', \hat{y}'') = \sqrt{\sum_{i=1}^{m} (\hat{y}'_i - \hat{y}''_i)^2},$$

and has a lower bound of $d_{min} = 0$ that is approached if and only if $\hat{y}' = \hat{y}''$. A normalized version of the ED is available through the *Gaussian radial basis function* (GRBF) kernel, that is commonly used in support vector machines to assess the proximity between two data points. It is defined as

$$s_{\text{GRBF}}(\hat{y}', \hat{y}''; \sigma) = \exp\left(-\frac{d_{\text{ED}}(\hat{y}', \hat{y}'')^2}{2\sigma^2}\right),$$

and can be interpreted as a similarity measure. σ is a free parameter that can be used to regulate the "sensitivity" of the measure, where smaller values lead to less similar predictions (i.e., $s_{\text{GRBF}}(\hat{y}', \hat{y}''; \sigma_1) < s_{\text{GRBF}}(\hat{y}', \hat{y}''; \sigma_2)$ when $\sigma_1 < \sigma_2$). Please note that, although this measure is normalized, it is not scale-invariant.

A scale-invariant option is the *concordance correlation coefficient* [CCC, Lin et al., 2002] that is a normalized version of the well-known *mean squared deviation* (MSD). It is defined as

$$s_{\text{CCC}}(\hat{y}', \hat{y}'') = \frac{2\sigma_{\hat{y}'\hat{y}''}}{\sigma^2_{\hat{y}'} + \sigma^2_{\hat{y}''} + (\mu_{\hat{y}'} - \mu_{\hat{y}''})^2},$$

and can be computed by plugging in the standard sample estimates for means, variances, and covariances. When using the divisor m (instead of $m - 1$) in the sample estimates, the values of the CCC are bounded within $[-1, 1]$, where the lower bound indicates perfect disagreement and the upper bound indicates perfect agreement between the predictions, respectively.

Other measures presented in the literature may also be used in our framework. A small selection of measures that we consider as useful for the regression case (without any claim to completeness) and have implemented in our R package is descibed in the supplementary material.

3.1.2 Classification

Let us now consider the classification case. A simple way to assess the similarity between two classification results is the *average class agreement* (ACA) between the predicted class labels, computed by

$$s_{\text{ACA}}(\hat{y}', \hat{y}'') = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\hat{y}'_i = \hat{y}''_i},$$

where $\mathbb{1}$ is the indicator function. By definition, it is normalized between 0 and 1. In case of unbalanced response classes, the *Kappa statistic* may be used to additionally account for their distribution [see, e.g., Kuhn and Johnson, 2013, chap. 11].

For a probabilistic classification result, however, much more detailed information is provided by the predicted class probabilities. Therefore, we suggest using a more precise similarity estimation via

$$s(\hat{\pi}', \hat{\pi}'') = 1 - \frac{1}{m} \sum_{i=1}^{m} \delta(\hat{\pi}'_i, \hat{\pi}''_i),$$

where $\delta(\cdot)$ is a distance measure between two discrete probability distributions [see, e.g., Cha, 2007, for a survey on distance measures between probability distributions].

A small selection that we consider as useful and computationally feasible distance measures for discrete probability distribution (without any claim to completeness) is again provided in the supplementary material. A well-known version is the *total variation distance* (TVD) that is computed via

$$\delta_{\text{TVD}}(\hat{\pi}'_i, \hat{\pi}''_i) = \frac{1}{2} \sum_{k=1}^{K} |\hat{\pi}'_{ik} - \hat{\pi}''_{ik}|.$$

Due to the l_1-norm, it is more sensitive to small changes in the predicted class probabilities compared to other distance measures.

3.2 Resampling and evaluation methods

Let us consider the real data problem where the DGP is unknown. To assess the stability in this situation, two samples $\mathfrak{L}'_b$ and $\mathfrak{L}''_b$ for generating the results and a third sample $\mathfrak{E}_b$ for evaluating the similarity have to be generated in each iteration by resampling from the original data set $\mathfrak{L} = \{z_1, \ldots, z_n\}$ (as previously described in Section 2.2).

3.2.1 Learning overlap

An important aspect concerns the intersection between the learning samples, given by $\mathfrak{L}'_b \cap \mathfrak{L}''_b$, that we call the *learning overlap* (as opposed to the *evaluation overlap*, see below).

Naturally, the similarity of the predictions from two results depends on the similarity of the observations in the learning samples that are used to generate it [see, e.g., Ntoutsi et al., 2008]. It is to be expected, however, that the similarity of the predictions is additionally affected when observations are sampled into both learning samples, that is, when $\mathfrak{L}' \cap \mathfrak{L}'' \neq \emptyset$. One can think of the jackknife approach as the most extreme form of learning overlap, since except for a single observation, both samples share the same observations. In this case, the focus would rest on the *robustness* of a result with respect to small changes in the given learning sample. With less learning overlap, the focus shifts towards the *generalizability* of a result for independent draws from the DGP.

The learning overlap varies between the following exemplary resampling methods:

- Bootstrap sampling: We refer to bootstrap sampling as randomly drawing r observations with replacement from $\mathfrak{L}$, which is done twice per iteration to generate two

learning samples. The probability that observation z_i appears in both samples can be shown to be $\left[1-(1-\frac{1}{n})^r\right]^2$. If $r = n$, the expected size of the overlap between $\mathfrak{L}'_b$ and $\mathfrak{L}''_b$ is $\approx 40\%$ of the size of $\mathfrak{L}$ and decreases if $r < n$. Thus, the overlap can be reduced by choosing $r < n$.

- Subsampling: We refer to subsampling as randomly drawing r observations without replacement from $\mathfrak{L}$, which is done twice per iteration to generate two learning samples. The probability that observation z_i appears in both samples is given by $\left[\frac{r}{n}\right]^2$ and is obviously 1 if $r = n$. For all r, subsampling generates a larger learning overlap than bootstrap sampling.

- Splithalf: We refer to splithalf sampling as splitting the learning sample into two disjoint sets of observations. To generate $\mathfrak{L}'_b$, sample $\lfloor \frac{n}{2} \rfloor$ observations without replacement from $\mathfrak{L}$. Then, let $\mathfrak{L}''_b = \mathfrak{L} \setminus \mathfrak{L}'_b$. The overlap is zero by definition. However, at the same time, the size of the learning samples is halved.

Hence, the commonly used resampling methods differ with respect to the sample size and the overlap of the learning samples.

3.2.2 Evaluation overlap

Another aspect concerns the sample $\mathfrak{E}_b$ used for evaluating the predictions and its overlap with the learning samples, given by $\mathfrak{E}_b \cap \{\mathfrak{L}'_b \cup \mathfrak{L}''_b\}$, that we call the *evaluation overlap*. It corresponds to the set of observations in $\mathfrak{E}_b$ previously used for generating either result. This is an important issue when the aim is prediction error estimation: In order to avoid overfitting, the learning and test sample should not overlap and there is always a tradeoff between keeping enough observations in the learning sample for precisely estimating the model versus reserving enough observations as a test sample for precisely estimating the prediction accuracy [see, e.g., Molinaro et al., 2005]. Since in our framework for stability assessment, however, we do not compare the predictions with the true response but only two predictions with each other, the evaluation overlap (related to the detectability of overfitting) is less of an issue, as will be illustrated in Section 4.3.2.

The evaluation overlap varies between the following exemplary evaluation methods:

- In-sample (ALL): The complete set of observations available is used for evaluating the similarity (i.e., $\mathfrak{E}_b = \mathfrak{L}$). How many of the observations in this sample were

previously used for generating the results depends on the resampling method.

- Out-of-bag (OOB): The observations not used for learning either of the two results are used for evaluating the similarity (i.e., $\mathfrak{E}_b = \mathfrak{L} \setminus \{\mathfrak{L}'_b \cap \mathfrak{L}''_b\}$). Hence, none of the observations in the evaluation sample was also present in any of the learning samples.

- Out-of-sample (OOS): A completely separate set of observations is used for evaluating the distance (i.e., $\mathfrak{E}_b \cap \mathfrak{L} = \emptyset$), which could result from separating the original data set into a learning and a test sample. None of the observations used in the evaluation sample was used for generating any of the results in the entire stability assessment procedure. Note that with this method, fewer observations are available for generating the results in the first place.

Hence, the common evaluation methods differ with respect to the sample size and the overlap of the evaluation sample with the learning samples.

The impact of the different combinations of resampling and evaluation methods will be illustrated by the simulation experiments presented in Section 4.3. The most strict way to investigate the practically relevant question of how dissimilar the interpretation of a result could be when a different learning sample was used for training, is when there is no learning and evaluation overlap and when the size of the learning and the evaluation samples remain equal to the size of the original data set. Later on, we therefore consider the stability assessed by means of independent draws from a known DGP (as carried out in the simulation study problem) as the "reference stability"; that is, as the gold standard. In situations where the DGP is unknown, we aim to come as close as possible to this reference.

3.2.3 Reweighting

A computationally efficient way for implementing the resampling in practice is by reweighting the observations in the original data set $\mathfrak{L}$. Instead of generating the learning samples $\mathfrak{L}'_b$ and $\mathfrak{L}''_b$ by resampling from the original data set, the n-dimensional vectors $w'_b = \{w'_{ib}\}$ and $w''_b = \{w''_{ib}\}$ $(i = 1, \ldots, n)$ that contain nonnegative case-weights are defined. The case-weights of observation i are given by

$$w'_{ib} = \#\{i : x_i \in \mathfrak{L}'\} \quad \text{and} \quad w''_{ib} = \#\{i : x_i \in \mathfrak{L}''\}.$$

By means of these case-weights, all common resampling schemes can be represented.

For example, the case-weights $w'_{ib}, w''_{ib} \in \{0, 1\}$ apply for subsampling and splithalf sampling and the case-weights $w'_{ib}, w''_{ib} \in \{0, 1, 2, 3, \ldots\}$ apply for bootstrap sampling.

The results $r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}, w'_b)$ and $r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}, w''_b)$ are then generated using w'_b and w''_b, respectively, and the predictions are given by

$$\hat{y}' = \{r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}, w'_b) : x \in \mathfrak{L}\} \quad \text{and} \quad \hat{y}'' = \{r_{\mathcal{A},\mathcal{M}}(x; \mathfrak{L}, w''_b) : x \in \mathfrak{L}\}.$$

However, whether case-weights can be applied in practice depends on the algorithm and its software implementation.

To implement a specific evaluation method or restrict the analysis to a particular region of the predictor space, one can define $w^e_b = \{w^e_{ib}\}$ $(i = 1, \ldots, n)$, that is also a n-dimensional vector with nonnegative case-weights. The predictions used for estimating the similarity $s(\hat{y}'_b, \hat{y}''_b)$ are now selected from $\hat{y}'$ and $\hat{y}''$ according to the counts in w^e_b.

4 Simulation and benchmark experiments

This section illustrates main points of the conceptual contribution and lends empirical support to it. We first describe the architecture of the DGPs used in the simulation experiments. Then, we present the results of two simulation studies. In the first study, we analyze the effect of different characteristics of the DGP on the stability of a result and relate that to our claim that the stability of a result is a property of both the algorithm and the DGP. In the second study, we analyze different combinations of resampling and evaluation methods for stability assessment to show that a larger learning overlap tends to produce higher similarity values. We end with a benchmark experiment to illustrate how the framework can be applied in practice.

We restrict our analysis to binary classification problems and two popular examples of an unstable and stable method, recursive partitioning and logistic regression, respectively. We use the following implementations available in R: The function **ctree()** from the **partykit** package for conditional inference trees and the function **glm()** with `family = "binomial"` for logistic regression. For both methods, the default settings were used.

4.1 Data-generating processes

Here, the aim is to investigate the impact of several characteristics of the DGP on the stability assessment. In a preliminary study (results not shown for brevity) we observed

notable impact of the learning sample size, the distribution of the response, and the dimension of the predictor space. Further, an important conceptual part of the framework is the emphasis on the match between algorithm and DGP for stability, so we investigate different forms for f. Therefore, the DGPs are set up as follows:

Dimensionality For all DGPs, the predictor variables are sampled from a multivariate standard normal distribution, $x_i \sim \mathcal{N}_p(0, I_p)$, where p is the number of predictor variables comprising $q \leq p$ signal and $p - q$ noise variables. We use $p = 20$ and $p = 40$, with $q = 4$ and $q = 8$ signal variables, respectively.

Functional form The binary response is sampled from a Bernoulli distribution with a conditional probability of success given by

$$\pi_i = \mathsf{P}(Y_i = 1|x_i) = \text{logit}^{-1}\left(\beta_0 + \sum_{j=1}^{q} f(x_{ij})\right).$$

The intercept β_0 specifies the baseline probability and was used to control the class distribution (see below). For $f(x)$ we use either $f(x) = x$ or $f(x) = \text{sgn}(x)$. The choices of $f(x)$ are motivated by the algorithms considered for the simulations so that they match the DGPs: logistic regression can model linear effects (identity function), trees model piecewise constant effects (signum function).

Sample size Different sample sizes are selected according to the binary logarithm (for easy plotting):

$$n = 2^7(= 128),\ 2^8(= 256),\ \ldots,\ 2^{15}(= 32\,768).$$

Class distribution Three different class distributions are investigated for the response: Equally balanced classes (50%/50%), weakly unbalanced classes (30%/70%) and strongly unbalanced classes (10%/90%). To approximately achieve these allocations, the intercept β_0 is chosen at 0, 1.417, and 3.447 for $p = 20$ and at 0, 1.774, and 4.315 for $p = 40$.

4.2 Study 1: Impact of the DGP

4.2.1 Reference stability

Here we investigate the reference stability of results generated by training an algorithm on data from a DGP that it can learn well (they match) by varying the learning sample size, the

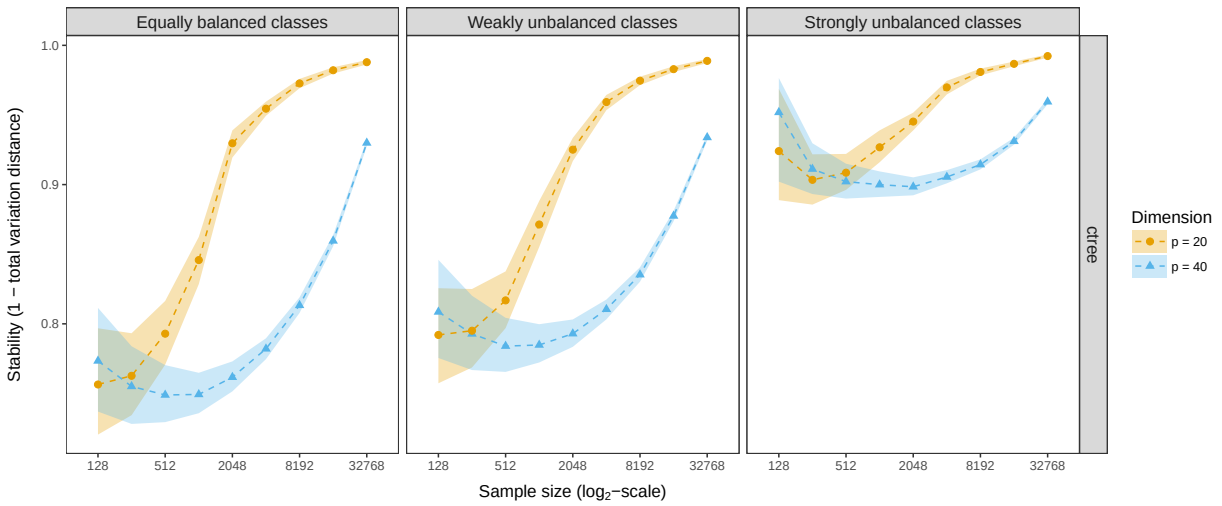

Figure 2: Stability assessment with the total variation distance of results generated by training the algorithm **ctree** on DGPs based on the signum function with different sample sizes, distribution of the response classes and dimension of the predictor space. The dashed line marks the 50%-quantile of the estimated similarity distribution (uncertainty bands illustrate the 25%- and the 75%-quantiles).

dimension of the predictor space, the class distribution, and the functional form. We limit ourselves to the results for **ctree**; the results for **glm** can be found in the supplementary material.

Figure 2 shows the results. The panels separate the three different class distributions, the two dimensions of the predictor space are displayed in different colors, and the sample sizes are depicted on the x axis. The similarity measure used was based on the total variation distance (1-TVD; see Section 3.1.2). To estimate the similarity distribution P_S precisely, the procedure was repeated $B = 5000$ times (in practice fewer iterations are sufficient). The dashed lines in Figure 2 correspond to the median similarity, the uncertainty bands are the range between the lower and upper quartile of P_S for each sample size.

As one can see, the results become more stable as the learning sample size increases. For the DGPs with $p = 20$ (orange) the similarity values approach the upper bound of the similarity measure (to almost perfectly stable). For the DGPs with $p = 40$ (blue) this convergence is slower and does not reach the upper bound for the examined sample sizes. For both scenarios, the variation of the similarity values decreases as the sample size increases.

For the case of $p = 40$, the median similarity values display a "U-shape" as a function of the sample size. This is caused by the property that trees with fewer splits (from small samples) are generally more stable than trees with more splits (medium-sized samples).

Tree results eventually become more stable again due to the increased learning sample size. This also highlights that high stability can go hand in hand with lower prediction accuracy (as is the case for smaller trees), so that a measure of stability can give additional information not captured by accuracy. We will return to this aspect in the application example in Section 4.4.

The stability is smaller for DGPs with balanced classes than for DGPs with unbalanced classes, which can be explained by the smaller range of values from which the conditional probabilities are sampled in the unbalanced case. Thus, any differences between the predictions of the results become smaller and the stability correctly increases.

4.2.2 (Mis-)match between DGP and algorithm

Here, we investigate how the match – or a lack thereof – between the DGP and the algorithm affects the stability assessment of a result. We restrict ourselves to equally balanced classes and $p = 20$. Additional conditions can be found in the supplementary material: the conclusions are qualitatively the same.

The results are illustrated in Figure 3 (with medians as well as the lower and the upper quartiles of the similarity distributions). The left panel shows results for **ctree**, the right panel for **glm**. The two different DGPs are depicted in orange (identity function) and blue (signum function). Thus in the left panel the DGP with the signum function is the better match, in the right panel it is the identity function.

We find that, while the difference is much less prominent for the **glm**, both algorithms show more stable results on average for the matching DGP. This supports our conceptualisation of stability as a property of both the algorithm and the functional form of the DGP.

4.3 Study 2: Impact of resampling and evaluation methods

4.3.1 Common choices for resampling and evaluation methods

Here we investigate the impact of different combinations of the commonly used resampling and evaluation methods (see Table 1) using the following procedure for each algorithm and DGP:

1. First, we assessed the reference stability for the selected algorithm and the DGP and computed the mean of the corresponding similarity distribution, denoted by $\bar{s}_0$.

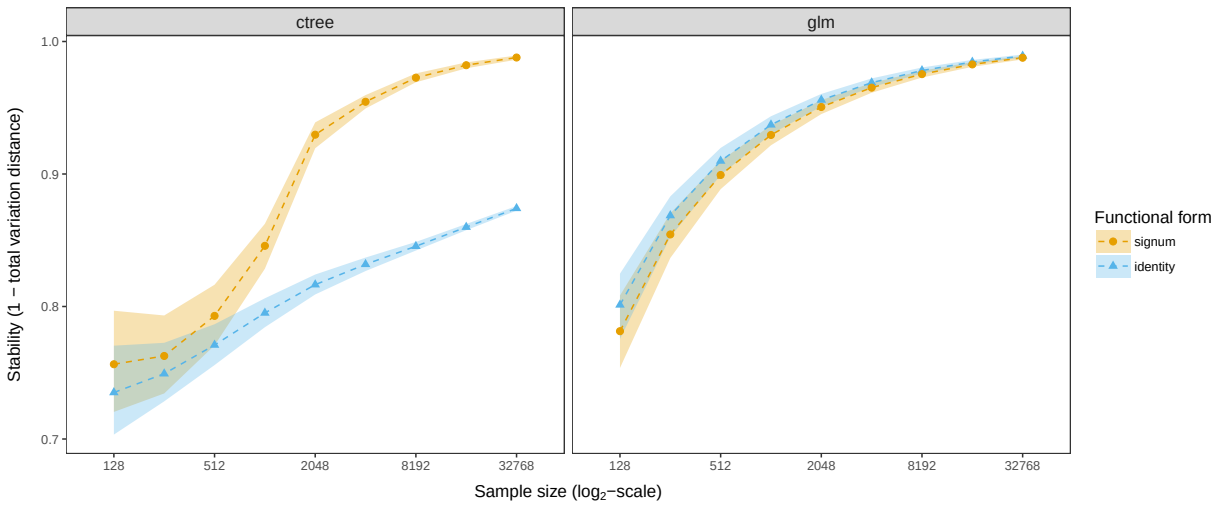

Figure 3: Stability assessment of DGPs with different functional forms (signum and identity) and two algorithms (**ctree** and **glm**) for different sample sizes. The dashed line corresponds to the 50%-quantile of the estimated similarity distribution (uncertainty bands illustrate the 25%- and the 75%-quantiles).

2. For $l = 1, \ldots, 100$ we repeated the following steps:

   (a) Draw a learning sample $\mathfrak{L}$ from the DGP and generate the result $r_a(x; \mathfrak{L})$.

   (b) Assess the stability of $r_a(x; \mathfrak{L})$ as described in Section 2.3 with $B = 500$ using the resampling combinations listed in Table 1.

   (c) For each combination, compute the mean of the corresponding similarity distribution, denoted by $\bar{s}_l$.

   (d) Compute the mean difference between $\bar{s}_l - \bar{s}_0$.

Note that this is not the recommended procedure when the DGP is known, as described in Section 2.3, but gives us the possibility to assess and compare different resampling and evaluation methods for situations where the DGP is unknown.

The estimated 100 mean differences to the reference stability for each combination are illustrated by means of boxplots in Figure 4. The learning and the evaluation overlap as well as the effective sample size for learning and evaluation differ between the combinations (see Table 1). Negative values imply that the mean of the reference stability was underestimated. Results are shown here only for DGPs with equally balanced response classes, $p = 20$, and a match of DGP and algorithm. The results for the remaining conditions can be found in the supplementary material.

Table 1: Resampling and evaluation methods used in the simulation study. Columns five and six show the expected size of the learning sample and the expected learning overlap; the columns seven and eight show the expected size of the evaluation sample and the expected evaluation overlap. The values are given relative to the original sample size n or to the size of the learning samples r. w denotes the proportion of observations hold out over all repetitions for evaluation. v denotes the proportion of observations drawn from the observations available for learning given by $(1 - w)n$.

Resampling	v	Evaluation	w	Learning $\mathfrak{L}', \mathfrak{L}''$		Evaluation $\mathfrak{E}$	
				$\frac{\lvert\mathfrak{L}'\rvert}{n} = \frac{\lvert\mathfrak{L}''\rvert}{n}$	$\frac{\lvert\mathfrak{L}' \cap \mathfrak{L}''\rvert}{n}$	$\frac{\lvert\mathfrak{E}\rvert}{n}$	$\frac{\lvert\mathfrak{E} \cap \{\mathfrak{L}' \cup \mathfrak{L}''\}\rvert}{r}$
Bootstrap sampling	1.0	Out-of-bag	0.0	100 %	≈ 40 %	≈ 13.5 %	0 %
Bootstrap sampling	1.0	In-sample	0.0	100 %	≈ 40 %	100 %	≈ 86.5 %
Bootstrap sampling	1.0	Out-of-sample	0.25	75 %	≈ 30 %	25 %	0 %
Bootstrap sampling	0.9	Out-of-bag	0.0	90 %	≈ 35.2 %	≈ 16.5 %	0 %
Bootstrap sampling	0.9	In-sample	0.0	90 %	≈ 35.2 %	100 %	≈ 83.5 %
Bootstrap sampling	0.9	Out-of-sample	0.25	67.5 %	≈ 26.2 %	≈ 22.5 %	0 %
Subsampling	0.8	Out-of-bag	0.0	80 %	≈ 64 %	≈ 4 %	0 %
Subsampling	0.8	In-sample	0.0	80 %	≈ 64 %	100 %	≈ 96.0 %
Subsampling	0.8	Out-of-sample	0.25	60 %	≈ 48 %	25 %	0 %
Splithalf sampling	0.5	In-sample	0.0	50 %	0 %	100 %	100 %
Splithalf sampling	0.5	Out-of-sample	0.25	37.5 %	0 %	25 %	0 %

In the upper row of Figure 4, the mean differences are illustrated for **ctree** and in the lower row for **glm**, respectively. The columns separate the three different samples sizes and the resampling combinations are illustrated in the x direction of the graph. We illustrate it for $n = 128$, 512, and 2048.

For **glm** (see lower row) we observe that the absolute stability difference decreases with a higher learning overlap: *half* < *boot.9* < *boot1* < *subs.8*. It also decreases with increasing size of the evaluation sample: *oob* > *oos* within the same resampling method. Finally, the stability difference remains almost constant for increasing evaluation overlap: *oob* ≈ *all* within the same resampling method. The mean reference stability $\bar{s}_0$ is captured well by bootstrap sampling (*boot*) with in-sample (*all*) or out-of-bag (*oob*) evaluation. With splithalf sampling (*half*) and subsampling (*subs*) the reference stability is under- or overestimated within the investigated sample sizes.

For **ctree** (see upper row), the results are less clear and none of the combinations approached the mean reference stability $\bar{s}_0$ accurately over all sample sizes. This might be

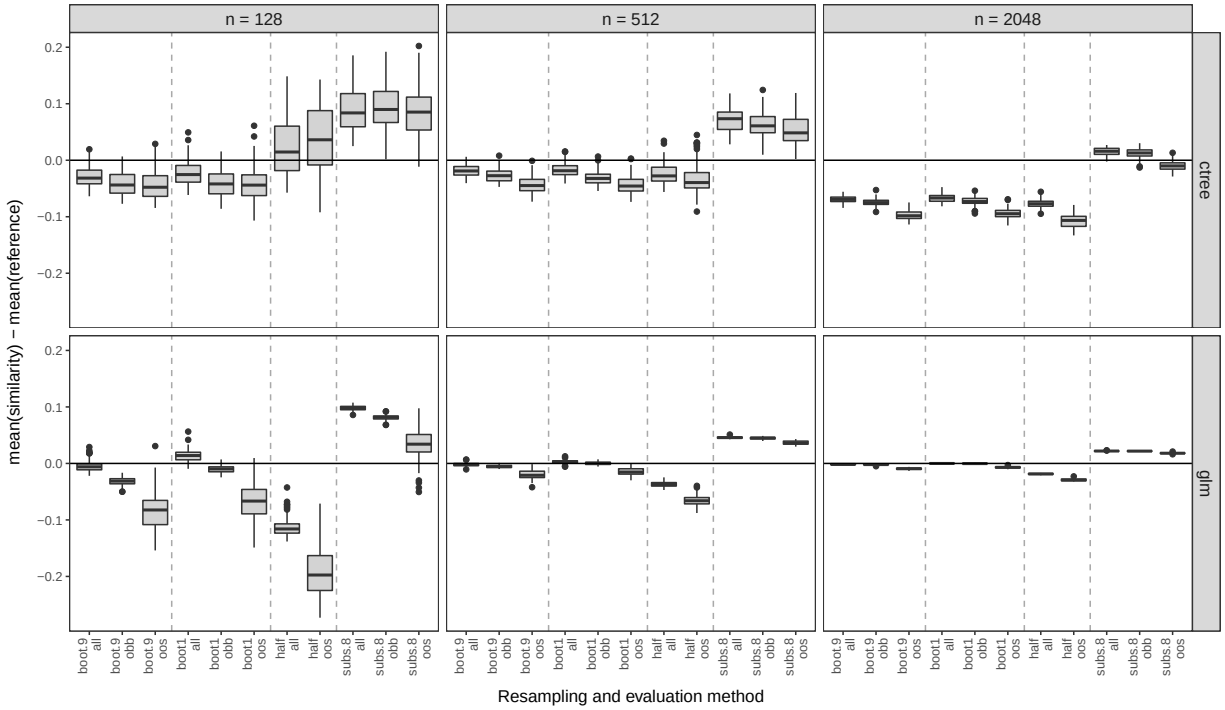

Figure 4: Boxplots of 100 differences between the mean of the reference similarity distribution and the mean of similarity distributions generated with different combinations of resampling (boot.9 = bootstrap sampling with $v = 0.9$, boot1 = bootstrap sampling with $v = 1$, half = splithalf sampling, subs.8 = subsampling with $v = 0.8$) and evaluation methods (all = in-sample, oob = out-of-bag, oos = out-of-sample). The upper and the lower row represent the results for ctree (for DGPs with signum function) and glm (for DGPs with identity function), respectively.

due to a confounding between the properties of the resampling combination and the trees' tuning parameters.

4.3.2 Learning and evaluation overlap

The resampling combinations of the previous section, although popular and feasible, do not allow to single out individual effects of the learning and the evaluation overlap (see Table 1). Here we demonstrate that the stability is largely affected by the learning overlap, but less by the evaluation overlap. For this we conducted a controlled computer experiment in which each overlap was systematically and individually manipulated to assess its partial impact. We only performed this analysis for DGPs with the lower dimensionality ($p = 20$) and equally balanced classes. This setting does not correspond to any practically applicable resampling or evaluation method.

In each of $l = 1, \ldots, 100$ repetitions we first drew an original data set $\mathfrak{L}$ of size $n = 1536$

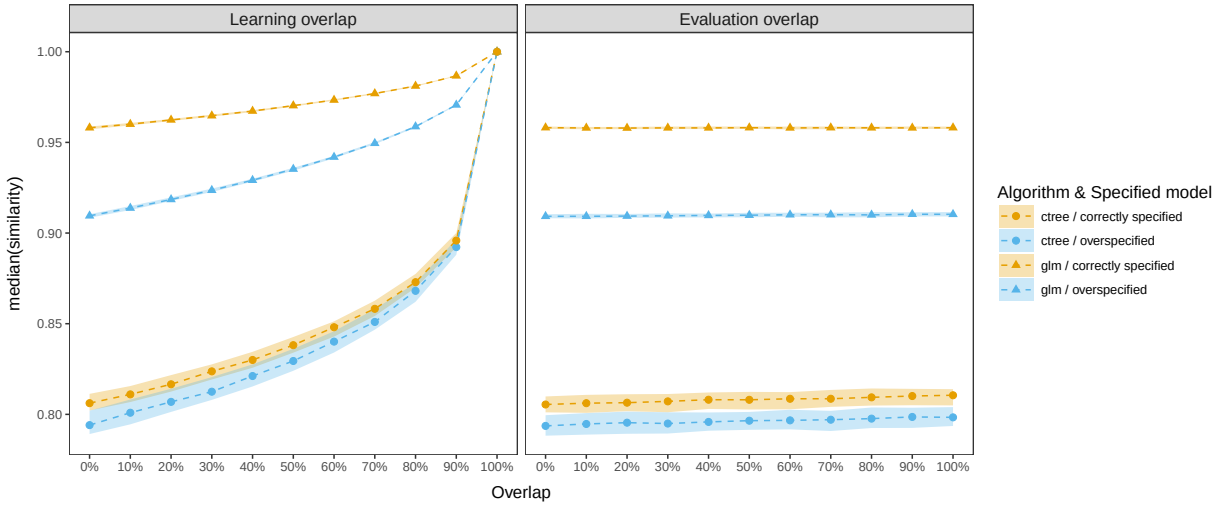

Figure 5: Stability assessment of results generated by experimentally manipulating the learning and the evaluation overlap. The dashed line corresponds to the 50%-quantile of $\bar{s}_l$ for **ctree** (cirlces) and **glm** (triangles) and the colors distinguish the specified models (uncertainy bands from the 25%- and the 75%-quantile).

($= 3 \times 512$) from the DGPs with the functional forms matching the algorithms **ctree** and **glm**. Next, the stability was assessed for two results: One generated by means of an overspecified model (using all predictor variables) and another generated by means of a correctly specified model (using only signal predictors). Then, in each of the $b = 1, \ldots, 500$ iterations of the stability assessment, we assembled the observations from the original data set $\mathfrak{L}$ into the three samples $\mathfrak{L}'_b$, $\mathfrak{L}''_b$ and $\mathfrak{E}$ of size $r = 512$, such that either the learning overlap or the evaluation overlap ranged from 0%, 10%, 20%, etc. to 100% (relative to the size of the learning and evaluation samples). Thus, in this study, the reference stability corresponded to the situation with 0% overlap. Finally, the mean similarity was computed in each repetition.

Thus, we ended up with a distribution of 100 mean values from which the median as well as the lower and the upper quartile are illustrated in Figure 5. The different overlaps are depicted on the x-axis. The colors orange and blue distinguish the correctly specified from the overspecified model.

We consider first the left panel, illustrating the results for the varying learning overlap, with 0% evaluation overlap. For both algorithms and models and with increasing overlap, the stability approaches the upper bound of the similarity measure (i.e., 1). Although the largest increase in stability is between 90% and 100%, we also observe substantial differences between 0% (corresponding to the overlap for splithalf sampling) and 80% (corresponding

to the overlap for subsampling).

In practice the learning overlap should be kept as low as possible in order to allow the strictest stability assessment. Too high an overlap leads to overly optimistic assessment (see subsampling). Splithalf, although without overlap, is not recommended as the effective learning sample size is much reduced, which also affects the stability assessment for small to medium-sized data sets. Bootstrap sampling retains a lot of the information in the learning sample with producing roughly 40% overlap. Bootstrap sampling with $r = v \cdot n$, $v < 1$ produces less overlap than standard bootstrap sampling with $v = 1$ while retaining roughly $v \cdot 100\%$ of the information. For values of v below but close to 1 this does not lead to substantially less accurate reference stability estimation than with $v = 1$, yet the overlap is strictly lower. This explains why bootstrap sampling with $v = 0.9$ performed best in most settings of the simulation study presented in Section 4.3.1.

The results also show that the stability is generally higher for **glm** (filled triangles) than for **ctree** (filled points). A new insight is that the stability is higher for a correctly specified model (illustrated in orange) than for an overspecified model with additional noise variables (illustrated in blue). This is true for both algorithms. Note, however, that the **ctree** algorithm shows overall lower stability but is less affected by noise variables due to its automatic variable selection.

For the evaluation overlap, illustrated in the right panel, the learning overlap was kept constant at zero. With increasing overlap, the stability remains almost constant for both algorithms and models. A small increase can be detected with **ctree** for both models and with **glm** for the overspecified model. According to preliminary results (not shown for brevity), this increase is slightly more pronounced for smaller samples sizes. We thus conclude, in accordance with our reasoning in Section 3.2.2, that the evaluation overlap itself has only little impact on the stability assessment.

4.4 Benchmark experiment

To illustrate the stability assessment in a practical scenario, we trained **ctree** on eight well-known benchmarking problems for classification from the UC Irvine meachine learning repository [Lichman, 2013] and the Titanic data from the R package **stablelearner**. The stability was assessed via $1-TVD$ with bootstrap sampling and out-of-bag evaluation using $B = 500$ iterations. We also assessed prediction accuracy via the *Kappa statistic* [see,e.g.,

Table 2: Median and IQR of stability and accuracy values for **ctree** on well-known classification problems. Results are listed in decreasing order of the median stability. Legend: n = sample size, p = number of predictors, K = number of classes.

Data set	n	p	K	Stability		Accuracy		CPU time [sec]
				Median	IQR	Median	IQR	
Iris	150	4	3	0.956	0.048	0.919	0.146	4.7
Breast Cancer	699	9	2	0.933	0.029	0.864	0.080	13.4
Titanic	1317	7	2	0.925	0.027	0.561	0.086	13.6
Ionosphere	351	34	2	0.900	0.058	0.788	0.132	10.8
Pima	768	8	2	0.835	0.036	0.403	0.130	13.8
Satellite	6435	36	6	0.819	0.013	0.363	0.025	631.3
Sonar	208	60	2	0.728	0.098	0.412	0.231	12.6
Vehicle	846	18	4	0.723	0.050	0.552	0.084	50.3
Glass	214	9	6	0.694	0.107	0.219	0.185	12.9

Kuhn and Johnson, 2013, chap. 11] (values 0/1 indicate no/perfect agreement between the true and the predicted class). A computer with a four core Intel i7-2600 processor running with 3.7 GHz in total and 8 GB RAM on a 64-bit Linux (Ubuntu 14.04.5 LTS) operating system was used to perform the analyzes. The results are given in Table 2. For each data set, the table lists the median and the interquartile range (IQR) of the stability and the accuracy values, as well as the CPU time in seconds and additional information about the data sets.

Generally, the results are more stable on data sets with fewer predictors and response classes. The result on the Ionosphere data set is a notable exception from this observation. For most problems, stability increased along with the accuracy. However, as mentioned earlier, there might be exceptions, like here, for example, for the Titanic and the Pima data set.

The results from this exercise can be used as a gauge for the reliability of our conclusions: For example, one would be relatively safe to interpret a tree generated by **ctree** for the Iris data set with a high stability, whereas for the Glass data set with a lower stability the results of a single tree should not be overinterpreted.

5 Discussion

5.1 Summary and recommendations

Stability is an important property for drawing reliable conclusions from a statistical learning result to avoid erroneous conclusions and errors in decision making. It is crucial therefore to be able to judge the stability of interpreted results, but many algorithms do not provide measures of stability.

In this article, we have presented a general framework to assess stability. It can be used in applications to assess the stability of a result for a given algorithm and data set or in benchmark experiments to either compare the stability of results generated by training different algorithms on a given data set or to compare the stability of results generated by training different data sets on a certain algorithm. One can also assess the stability of results for artificial DGPs in this framework.

The stability is assessed by repeatedly computing the similarity between two results generated with learning samples drawn directly from the DGP or by resampling from the original data set. The measure of similarity should compare two results by their predictions and not by their structure. A non-exhaustive list of exemplary measures was presented.

In laying out the framework of measuring the stability of the result of an algorithm trained on a specific learning sample, we have shown by reasoning and in a simulation experiment that the stability of a result is not only a property of the algorithm alone but also of the DGP that has generated the learning sample.

We studied the impact of different data characteristics (sample size, dimensionality, and class distribution) on the stability of results generated by logistic regression and conditional inference trees for binary classification. We observed a large impact on the stability for all investigated factors. This provides useful insights into the mechanisms of the stability assessment itself.

When assessing the stability of a result for a real data set with our framework, a resampling and an evaluation method must be selected. We have investigated different resampling and evaluation methods for logistic regression and conditional inference trees. Our investigations did not reveal one generally optimal resampling combination, but did highlight that framework-specific factors influence the stability assessment, particularly the learning overlap and the size of the data set. The best resampling or reweighting

scheme would have as little learning overlap as possible, the highest possible learning sample size and, although that is less important, as little evaluation overlap as possible. This combination is not generally feasible with resampling. A good combination of all three factors appears to be bootstrap resampling with out of bag evaluation. Future research will further investigate whether bootstrap sampling with a sample size slightly smaller than the original sample is the best approach.

Another practically relevant application of the framework is the comparison of stability assessments from different results. Such comparisons should only be conducted when either the response variables were measured on the same scale or when a scale-invariant similarity measure is used. For comparability, a user should report the similarity measure, the resampling and evaluation method, and the number or iterations they employed when presenting stability results.

5.2 Limitations and future research

We used pairwise comparison of results generated from different learning samples for similarity assessment [Lange et al., 2002, Turney, 1995, see also] and to estimate the distribution P_s. However, there are alternative ways to estimate P_s. One could, for example repeatedly compute the similarity between results generated on resampled data sets and the result generated on the original data set [see, e.g., Bar-hen et al., 2015]. Another option is to estimate P_s from the similarities between all possible pairs of results generated on resampled data sets [see, e.g., Lim and Yu, 2016]. Future research should compare the different approaches to reveal what properties they have in common.

Another important open question is how stability is connected to variability. For example, it can be shown that the expected variability over the predictor space is proportional to the expected similarity computed by the Euclidean distance [Zhang et al., 2012]. It is not clear whether such a connection exists in general, but if it does, it could be exploited to reduce the computational costs in assessing the (expected) stability.

In our framework, stability is a global characteristic over the complete predictor space. A result is, however, almost certainly more stable in certain areas of the predictor space than in others. It would be interesting to also be able to investigate and illustrate the local stability of results.

6 Implementation

An implementation of the stability measuring framework presented in this paper is available as a package for R [R Core Team, 2016] from `http://r-forge.r-project.org/projects/stablelearner/`. The function `stability()` implements the stability assessment procedure with bootstrap sampling and out-of-bag evaluation as the default resampling and evaluation method and with $B = 500$. Several similarity and dissimilarity measures for regression and classification are implemented. Stability can be assessed for one or more results generated by a few predefined algorithms (see `?LearnerList`), but new algorithms can be integrated by the user. Parallelization can be utilized with a convenience option for multicore computation based on **parallel** (for supported platforms). More detailed stability analyses based on descriptive measures and graphical illustrations can be conducted for results generated by tree-based algorithms via the function `stabletree()` as described in Philipp et al. [2016].

7 Acknowledgments

The authors acknowledge the University of Zurich S3IT: Service and Support for Science IT for providing the support and the computational resources that have contributed to the research results reported in this publication. URL: `http://www.s3it.uzh.ch`.

SUPPLEMENTARY MATERIAL

Appendix: Tables with similarity measures for classification and regression problems and figures containing the complete results from the simulation experiments presented in Section 4.2 and Section 4.3

References

Banerjee, M., Y. Ding, and A.-M. Noone (2012). Identifying representative trees from ensembles. *Statistics in Medicine 31*(15), 1601–1616.

Bar-hen, A., S. Gey, and J.-M. Poggi (2015). Influence measures for CART classification trees. *Journal of Classification 32*(1), 21–45.

Bousquet, O. and A. Elisseeff (2002). Stability and generalization. *Journal of Machine Learning Research 2*(3), 499–526.

Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *The Annals of Statistics 24*(6), 2350–2383.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science 16*(3), 199–231.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees*. Wadsworth statistics/probability series. Monterey, CA: Wadsworth.

Briand, B. B., G. R. Ducharme, V. Parache, and C. Mercat-Rommens (2009). A similarity measure to assess the stability of classification trees. *Computational Statistics & Data Analysis 53*(4), 1208–1217.

Cha, S.-h. (2007). Comprehensive survey on distance / similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences 1*(4), 300–307.

Hothorn, T., K. Hornik, and A. Zeileis (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics 15*(3), 651–674.

Hothorn, T., F. Leisch, A. Zeileis, and K. Hornik (2005). The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics 14*(3), 675–699.

Kuhn, M. and K. Johnson (2013). *Applied predictive modeling*. New York, NY: Springer.

Lange, T., M. L. Braun, V. Roth, and J. M. Buhmann (2002). Stability-based model selection. *Advances in Neural Information Processing Systems 15*, 617–624.

Lichman, M. (2013). UC irvine machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Lim, C. and B. Yu (2016). Estimation stability with cross validation. *Journal of Computational and Graphical Statistics 25*(2), 464–492.

Lin, L., A. S. Hedayat, B. Sinha, and M. Yang (2002). Statistical methods in assessing agreement. *Journal of the American Statistical Association 97*(457), 257–270.

Miglio, R. R. and G. Soffritti (2004). The comparison between classification trees through proximity measures. *Computational Statistics & Data Analysis 45*(3), 577–593.

Molinaro, A. M., R. Simon, and R. M. Pfeiffer (2005). Prediction error estimation: A comparison of resampling methods. *Bioinformatics 21*(15), 3301–3307.

Mukherjee, S., P. Niyogi, T. Poggio, and R. Rifkin (2006). Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics 25*(1-3), 161–193.

Ntoutsi, I., A. Kalousis, and Y. Theodoridis (2008). A general framework for estimating similarity of datasets and decision trees: Exploring semantic similarity of decision trees. In M. Zaki, H. Park, C. J. Apte, and K. Wang (Eds.), *Proceedings of the 2008 SIAM International Conference on Data Mining*, Philadelphia, PA, pp. 810–821. Society for Industrial and Applied Mathematics.

Philipp, M., A. Zeileis, and C. Strobl (2016). A toolkit for stability assessment of tree-based learners. In A. Colubi, A. Blanco, and C. Gatu (Eds.), *Proceedings of COMPSTAT 2016 – 22nd International Conference on Computational Statistics*, pp. 315–325. The International Statistical Institute/International Association for Statistical Computing.

Poggio, T., R. Rifkin, S. Mukherjee, and P. Niyogi (2004). General conditions for predictivity in learning theory. *Nature 428*, 419–422.

R Core Team (2016). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rusch, T. and A. Zeileis (2013). Gaining insight with recursive partitioning of generalized linear models. *Journal of Statistical Computation and Simulation 83*(7), 1301–1315.

Shannon, W. D. and D. Banks (1999). Combining classification trees using MLE. *Statistics in Medicine 18*(6), 727–740.

Shmueli, G. (2010). To explain or to predict? *Statistical Science 25*(3), 289–310.

Stodden, V. (2015). Reproducing statistical results. *Annual Review of Statistics and Its Application 2*(1), 1–19.

Strobl, C., J. Malley, and G. Tutz (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods 14*(4), 323–348.

Turney, P. (1995). Technical note: Bias and the quantification of stability. *Machine Learning 20*(1-2), 23–33.

Wasserman, L. (2004). *All of statistics: A concise course in statistical inference*. New York, NY: Springer.

Yu, B. (2013). Stability. *Bernoulli 19*(4), 1484–1500.

Zeileis, A., T. Hothorn, and K. Hornik (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics 17*(2), 492–514.

Zhang, Y., H. Wu, and L. Cheng (2012). Some new deformation formulas about variance and covariance. In *Proceedings of 2012 International Conference on Modelling, Identification and Control*, Piscataway, NJ, pp. 1042–1047. Institute of Electrical and Electronics Engineers.

SUPPLEMENTARY MATERIAL

Supplementary material to the manuscript "Measuring the Stability of Results from Supervised Statistical Learning".

A Similarity measures

- Table 1 lists similarity measures for the regression case.
- Table 2 lists similarity and distance measures for the classification case.

B Simulation experiments

- Figure 1 and Figure 2 illustrate the complete results from Study 1 presented and discussed in the manuscript.
- Figures 3-8 illustrate the complete results from Study 2 presented and discussed in the manuscript.

Table 1: Similarity measures for the regression case.

Name	Formula or definition	Range	Scale-invariance	R package
Euclidean distance	$d_{\mathrm{ED}}(\hat{y}', \hat{y}'') = \sqrt{\sum_{i=1}^{m} (\hat{y}'_i - \hat{y}''_i)^2}$	$[0, \infty]$	no	`edist()`
Mean squared distance	$d_{\mathrm{MSD}}(\hat{y}', \hat{y}'') = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}'_i - \hat{y}''_i)^2$	$[0, \infty]$	no	`msdist()`
Root mean squared distance	$d_{\mathrm{RMSD}}(\hat{y}', \hat{y}'') = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (\hat{y}'_i - \hat{y}''_i)^2}$	$[0, \infty]$	no	`rmsdist()`
Mean absolute distance	$d_{\mathrm{MSD}}(\hat{y}', \hat{y}'') = \frac{1}{m} \sum_{i=1}^{m} \lvert \hat{y}'_i - \hat{y}''_i \rvert$	$[0, \infty]$	no	`madist()`
Quantile of absolute distance	$d_{\mathrm{QAD}}(\hat{y}', \hat{y}''; p) = \mathsf{P}(\lvert \hat{y}' - \hat{y}'' \rvert \leq \kappa) = 1 - p$	$[0, \infty]$	no	`qadist()`
Coverage probability	$s_{\mathrm{CP}}(\hat{y}', \hat{y}''; \kappa) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\lvert \hat{y}'_i - \hat{y}''_i \rvert \leq \kappa}$	$[0, 1]$	no	`cprob()`
Gaussian radial basis function kernel	$s_{\mathrm{GRBF}}(\hat{y}', \hat{y}''; \sigma) = \exp\left(-\frac{d_{\mathrm{ED}}(\hat{y}', \hat{y}'')^2}{2\sigma^2} \right)$	$[0, 1]$	no	`rbfkernel()`
Tanimoto coefficient	$s_{\mathrm{TC}}(\hat{y}', \hat{y}'') = \frac{\sum_{i=1}^{m} \hat{y}'_i \hat{y}''_i}{\sum_{i=1}^{m} (\hat{y}'_i)^2 + \sum_{i=1}^{m} (\hat{y}''_i)^2 - \sum_{i=1}^{m} \hat{y}'_i \hat{y}''_i}$	$[-\frac{1}{3}, 1]$	no	`tanimoto()`
Cosine similarity	$s_{\mathrm{CS}}(\hat{y}', \hat{y}'') = \frac{\sum_{i=1}^{m} \hat{y}'_i \hat{y}''_i}{\sqrt{\sum_{i=1}^{m} (\hat{y}'_i)^2} \sqrt{\sum_{i=1}^{m} (\hat{y}''_i)^2}}$	$[-1, 1]$	no	`cosine()`
Concordance correlation coefficient	$s_{\mathrm{CCC}}(\hat{y}', \hat{y}'') = \frac{2\sigma_{\hat{y}'\hat{y}''}}{\sigma^2_{\hat{y}'} + \sigma^2_{\hat{y}''} + (\mu_{\hat{y}'} - \mu_{\hat{y}''})^2}$	$[-1, 1]$	yes	`ccc()`
Pearson correlation coefficient	$s_{\mathrm{PCC}}(\hat{y}', \hat{y}'') = \frac{\sum_{i=1}^{m} (\hat{y}'_i - \mu_{\hat{y}'}) (\hat{y}''_i - \mu_{\hat{y}''})}{\sqrt{\sum_{i=1}^{m} (\hat{y}'_i - \mu_{\hat{y}'})^2} \sqrt{\sum_{i=1}^{m} (\hat{y}''_i - \mu_{\hat{y}''})^2}}$	$[-1, 1]$	yes	`pcc()`

Note. m is the size of the evaluation sample and $\mathbb{1}$ denotes the indicator function.

Table 2: Similarity and distance measures for the classification case.

Name	Formula or definition	Range	R package
Average class agreement	$s_{\mathrm{ACA}}(\hat{y}', \hat{y}'') = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\hat{y}_i' = \hat{y}_i''}$	$[0, 1]$	`clagree()`
Cohen's kappa	$s_{\mathrm{CK}}(\hat{y}', \hat{y}'') = 1 - \frac{1 - s_{\mathrm{ACA}}(\hat{y}', \hat{y}'')}{1 - \frac{1}{m} \sum_{k \in \mathcal{K}} \sum_{i=1}^{m} \mathbb{1}_{\hat{y}_i' = k} \sum_{i=1}^{m} \mathbb{1}_{\hat{y}_i' = k}}$	$[0, 1]$	`ckappa()`
Bhattacharyya distance	$\delta_{\mathrm{BD}}(\hat{\pi}_i', \hat{\pi}_i'') = 1 - \sum_{k=1}^{K} \sqrt{\hat{\pi}_{ik}' \cdot \hat{\pi}_{ik}''}$	$[0, 1]$	`bdist()`
Total variation distance	$\delta_{\mathrm{TVD}}(\hat{\pi}_i', \hat{\pi}_i'') = \frac{1}{2} \sum_{k=1}^{K} \lvert \hat{\pi}_{ik}' - \hat{\pi}_{ik}'' \rvert$	$[0, 1]$	`tvdist()`
Hellinger distance	$\delta_{\mathrm{HD}}(\hat{\pi}_i', \hat{\pi}_i'') = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=1}^{K} \left(\sqrt{\hat{\pi}_{ik}'} - \sqrt{\hat{\pi}_{ik}''} \right)^2}$	$[0, 1]$	`hdist()`
Jensen-Shannon divergence	$\delta_{\mathrm{JSD}}(\hat{\pi}_i', \hat{\pi}_i'') = \frac{1}{2} \sum_{k=1}^{K} \left(\hat{\pi}_{ik}' \cdot \log \frac{\hat{\pi}_{ik}'}{m_{ik}} + \hat{\pi}_{ik}'' \cdot \log \frac{\hat{\pi}_{ik}''}{m_{ik}} \right), \; m_{ik} = \frac{1}{2} \left(\hat{\pi}_{ik}' + \hat{\pi}_{ik}'' \right)$	$[0, 1]$	`jsdist()`

Note. m is the size of the evaluation sample, $\mathbb{1}$ denotes the indicator function and $\mathcal{K}$ denotes the set of class labels with $|\mathcal{K}| = K$.

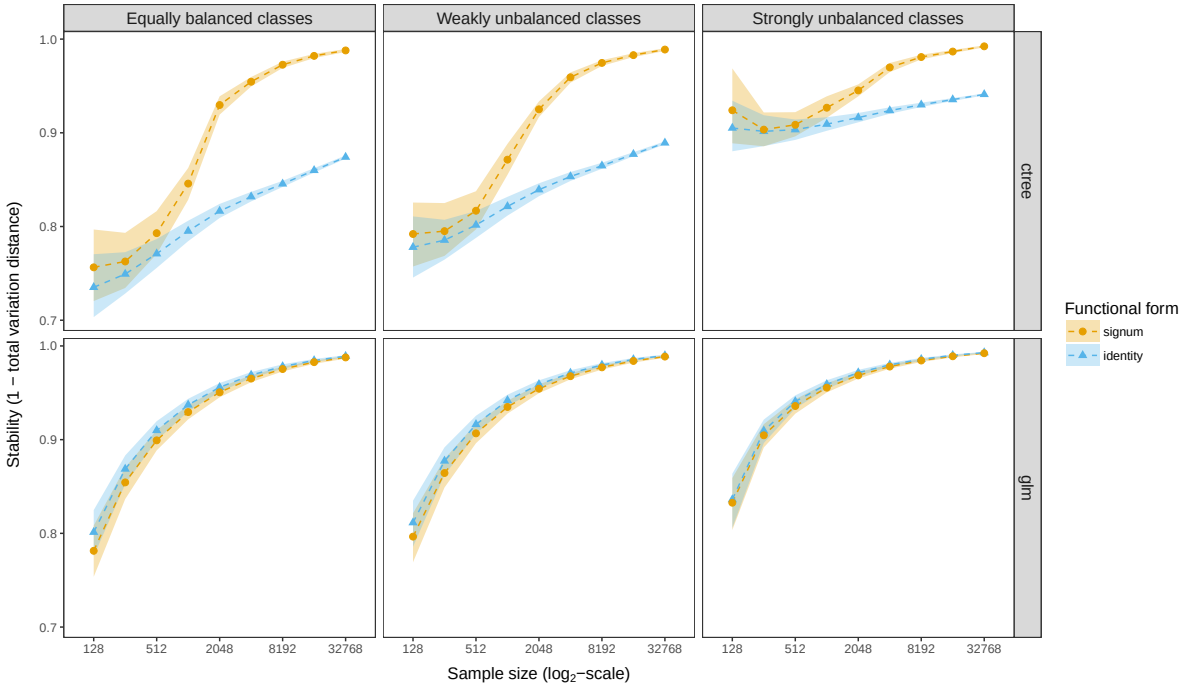

Figure 1: Reference stability (results of Study 1) for DGPs with low dimension ($p = 20$).

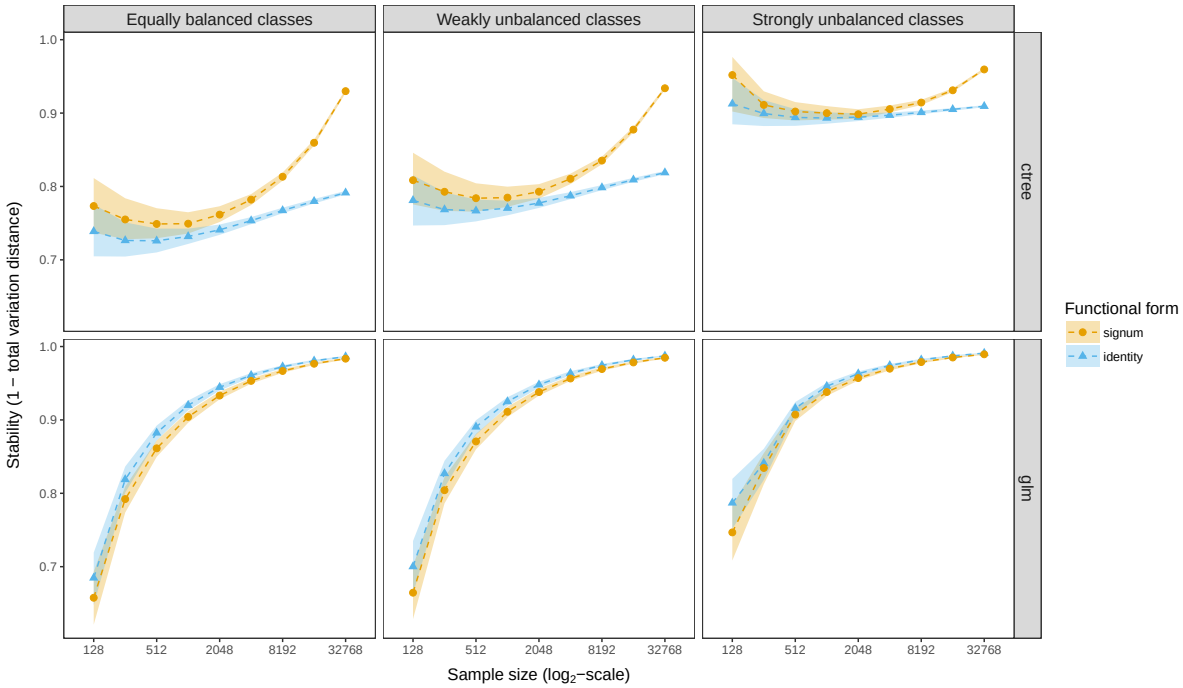

Figure 2: Reference stability (results of Study 1) for DGPs with high dimension ($p = 40$).

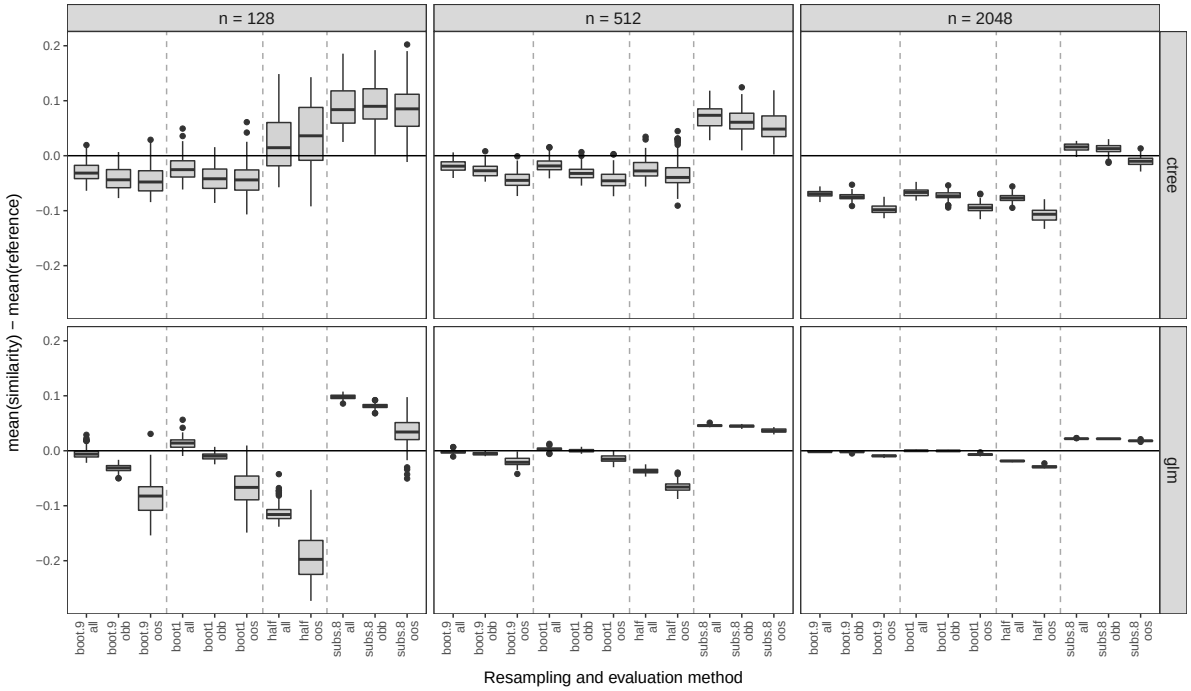

Figure 3: Results of Study 2 for DGPs with low dimension ($p = 20$) and equally balanced classes.

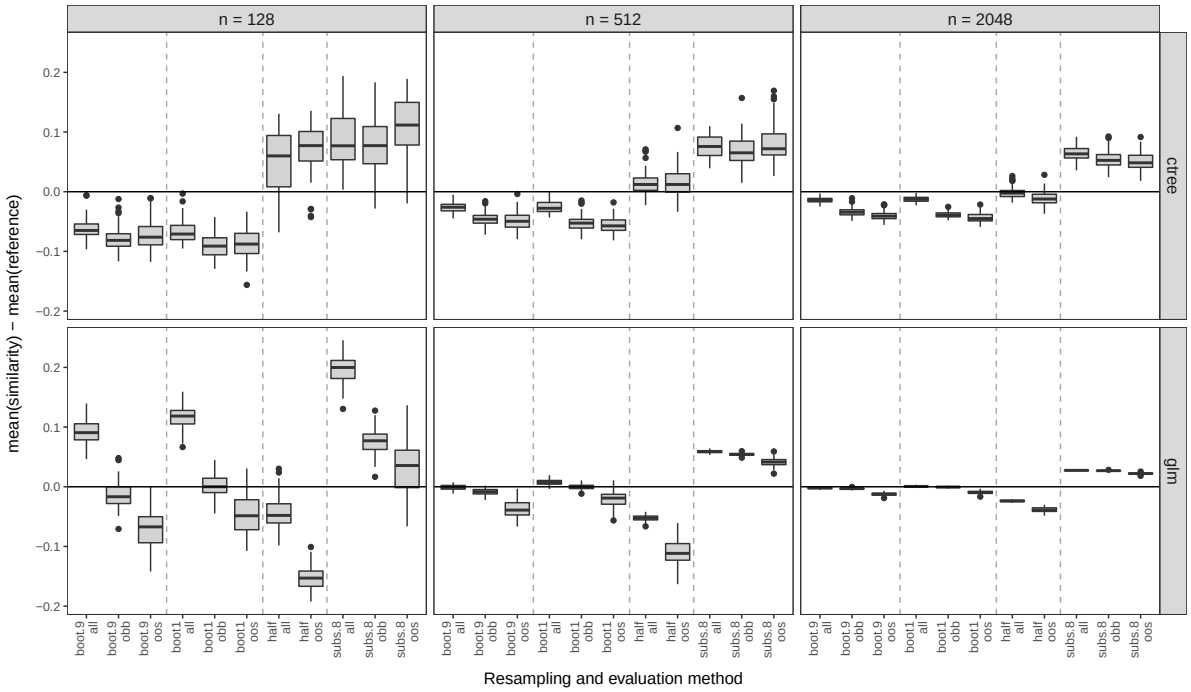

Figure 4: Results of Study 2 for DGPs with high dimension ($p = 40$) and equally balanced classes.

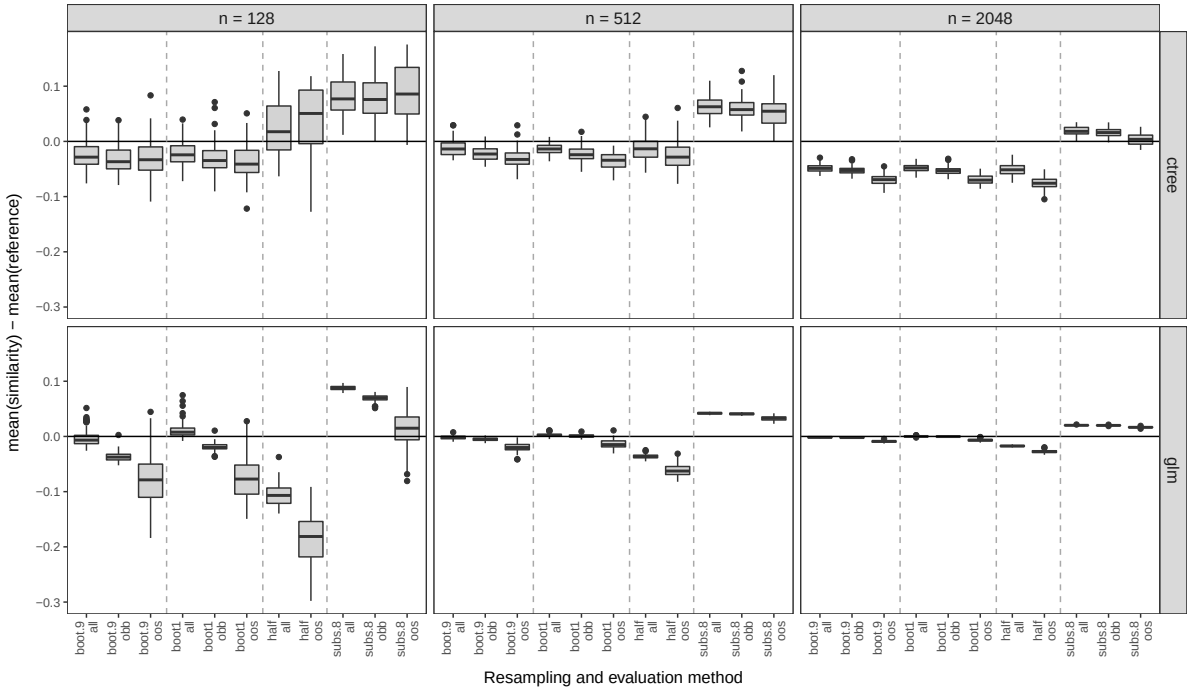

Figure 5: Results of Study 2 for DGPs with low dimension ($p = 20$) and weakly unbalanced classes.

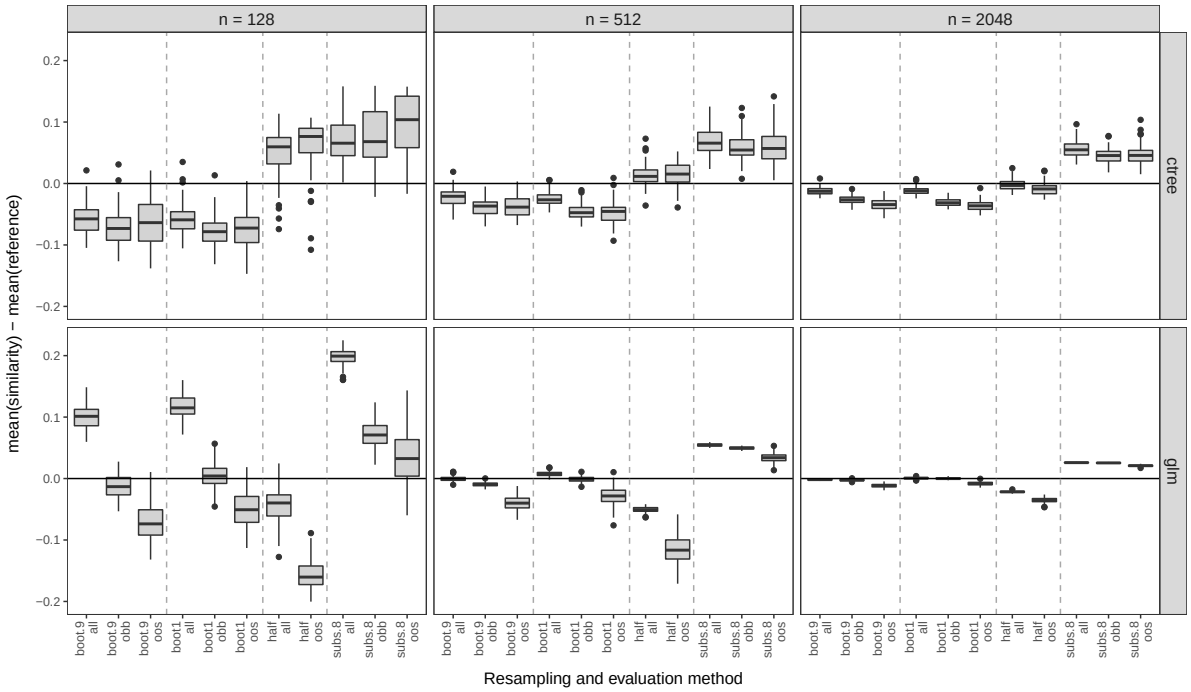

Figure 6: Results of Study 2 for DGPs with high dimension ($p = 40$) and weakly unbalanced classes.

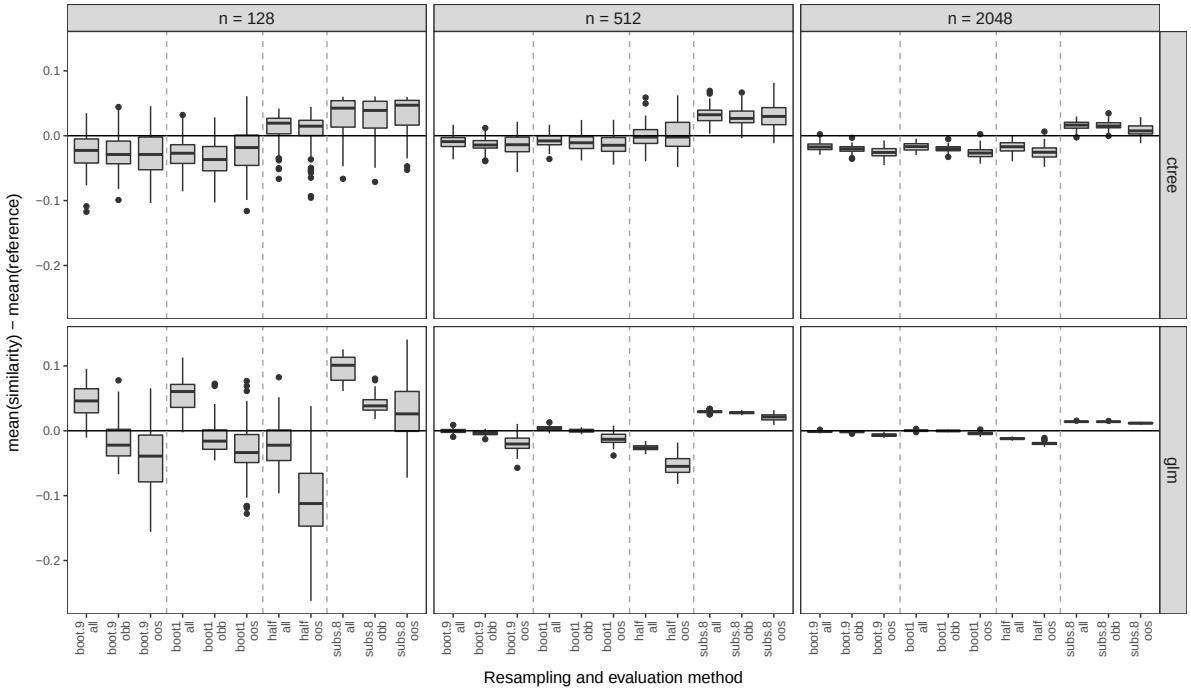

Figure 7: Results of Study 2 for DGPs with low dimension ($p = 20$) and highly unbalanced classes.

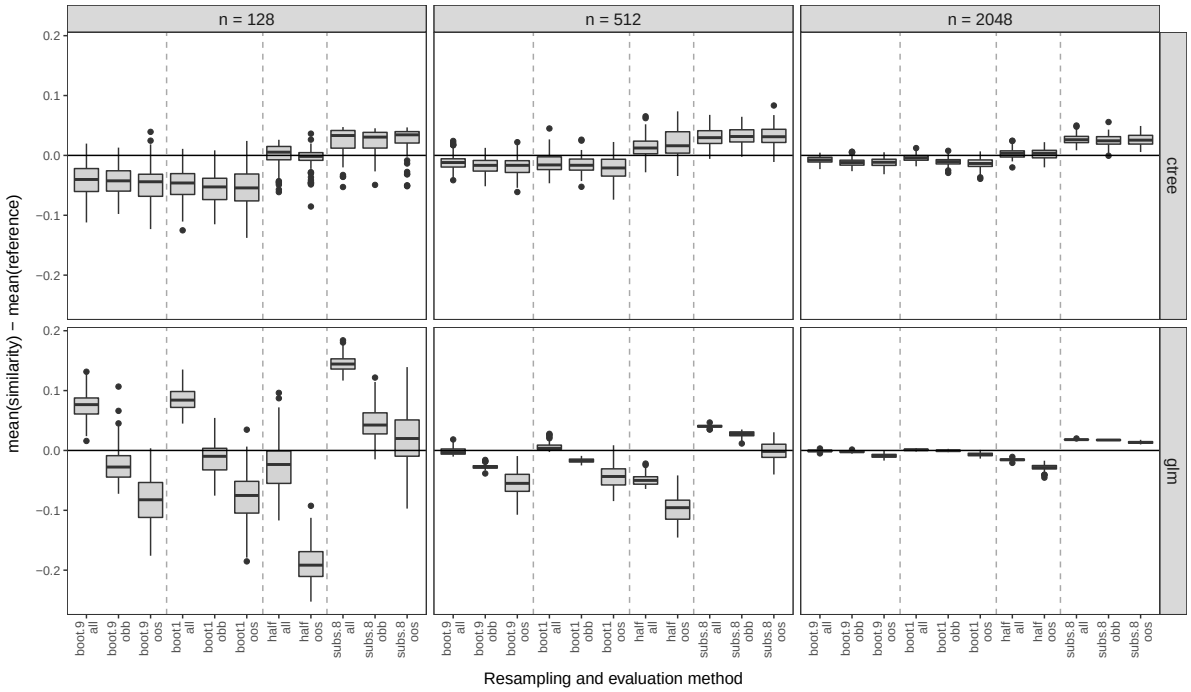

Figure 8: Results of Study 2 for DGPs with high dimension ($p = 40$) and highly unbalanced classes.