Journal Pre-proof

Nested cross-validation when selecting classifiers is overzealous for most practical applications

Jacques Wainer, Gavin Cawley

PII: S0957-4174(21)00654-0
DOI: https://doi.org/10.1016/j.eswa.2021.115222
Reference: ESWA 115222

To appear in: *Expert Systems With Applications*

Received date : 21 December 2019
Revised date : 16 March 2021
Accepted date : 14 May 2021

Please cite this article as: J. Wainer and G. Cawley, Nested cross-validation when selecting classifiers is overzealous for most practical applications. *Expert Systems With Applications* (2021), doi: https://doi.org/10.1016/j.eswa.2021.115222.

Nested cross-validation when selecting classifiers is overzealous for most practical applications

Jacques Wainer[a,*], Gavin Cawley[b]

[a]*Computing Institute*
University of Campinas
Campinas, SP, 13083-852, Brazil
[b]*School of Computing Sciences*
University of East Anglia
Norwich, NR4 7TJ, U.K.

Abstract

When selecting a classification algorithm to be applied to a particular problem, one has to simultaneously select the best algorithm for that dataset *and* the best set of hyperparameters for the chosen model. The usual approach is to apply a nested cross-validation procedure: hyperparameter selection is performed in the inner cross-validation, while the outer cross-validation computes an unbiased estimate of the expected accuracy of the algorithm *with cross-validation based hyperparameter tuning*. The alternative approach, which we shall call "flat cross-validation", uses a single cross-validation step both to select the optimal hyperparameter values and to provide an estimate of the expected accuracy of the algorithm that, while biased, may nevertheless still be used to select the best learning algorithm. We tested both procedures using 12 different algorithms on 115 real-life binary datasets and conclude that using the less computationally costly flat cross-validation procedure will generally result in the selection of an algorithm that is, for all practical purposes, of similar quality to that selected via nested cross-validation, provided the learning algorithms have relatively few hyperparameters to be optimised.

Key words: Hyperparameters; classification; cross-validation; nested cross-validation; model selection

1. Introduction

A practitioner who builds a classification model has to select the best algorithm for that particular problem. There are hundreds of classification algorithms described in the literature, such as k-nearest neighbour (Dasarathy,

*Corresponding author
Email addresses: wainer@ic.unicamp.br (Jacques Wainer), G.Cawley@uea.ac.uk (Gavin Cawley)

1991), SVM (Cortes and Vapnik, 1995), neural networks (Bishop, 1995), naïve Bayes (Hand and Yu, 2001), gradient boosting machines (Friedman, 2001), and so on. Although there are sometimes theoretical and/or empirical reasons to prefer a particular algorithm over another when tackling a specific problem, our current understanding of machine learning does not allow us to predict *a-priori* whether one algorithm will perform better than another. Furthermore, the so-called "no-free-lunch" theorems even suggest that no algorithm can outperform all others for all problems (Wolpert, 1996). Therefore, for most difficult tasks, one should benefit from trying many competing algorithms to discover which gives the best performance. However, most algorithms have one or more hyperparameters that must be set externally, for example, the k-nearest neighbour method has (usually) one hyperparameter, k, whereas random forest has at least two, the number of trees to be constructed and the number of features considered at each split. Unfortunately selecting an algorithm and tuning its hyperparameters are dependent steps: an algorithm may perform very well for a problem when using a particular set of hyperparameters, but may perform worse than other algorithms with a different, sub-optimal, set of hyperparameters. One will, therefore, want to choose both the algorithm and its hyperparameters in such a way as to maximize its expected performance on future data.

Choosing an appropriate model and optimising the hyperparameters are most often performed by minimising a cross-validation (Stone, 1974) estimate of generalisation performance. The most basic form of cross-validation (CV), known as *k-fold cross-validation* partitions the available data into k disjoint chunks of approximately equal size. In each iteration a training set is formed from a different combination of $k-1$ chunks, with the remaining chunk used as the test set; a model is then fitted to the training set and its performance evaluated using the test set. The average of the performance metric on the test set in each iteration is then used as an estimate of the generalisation performance of a model fitted to all of the available data. There are two common procedures for selecting the best algorithm and tuning the hyperparameters via cross-validation, the first is called *nested cross-validation*, also known as *double cross-validation* (Stone, 1974), the second appears to have no standard name, so we will call it *flat cross-validation*:

Flat Cross-Validation: The hyperparameters of each model are tuned to minimise a cross-validation based estimate of generalisation performance. The cross-validation performance estimate, evaluated for those optimal hyperparameter values, is then used to select the best model to use in operation. This approach is computationally inexpensive; however, an optimistic bias is introduced into the performance estimate as it has been directly optimised in tuning the hyperparameters (Cawley and Talbot, 2010). Unless this bias is commensurate for all of the candidate models, the re-use of the hyperparameter optimisation criterion as a model selection criterion may result in a sub-optimal choice of model, potentially selecting a model that is particularly susceptible to this bias, rather than a model with genuinely higher performance.

Nested Cross-Validation: An outer cross-validation procedure is performed to provide a performance estimate used to select the optimal model. In each fold of the outer cross-validation, the hyperparameters of the model are tuned independently to minimise an inner cross-validation estimate of generalisation performance. The outer cross-validation is then essentially estimating the performance of a method for fitting a model, emphincluding cross-validation based hyperparameter tuning. This eliminates the bias introduced by the flat cross-validation procedure as the test data in each iteration of the outer cross-validation has not been used to optimise the performance of the model in any way, and may, therefore, provide a more reliable criterion for choosing the best model. The computational expense of nested cross-validation, however, is substantially higher.

This study aims to perform an empirical evaluation to determine whether the additional computational expense of the nested cross-validation procedure is generally justified by providing a more reliable means of choosing the best model and statistically superior performance.

In practice, using flat cross-validation to select both the algorithm and the hyperparameters has already been in widespread use, due to the computational expense of nested cross-validation. The research area usually known as AutoML deals with automatically searching for the best combination of hyperparameters and algorithms for a particular classification (or regression) problem (Feurer et al., 2015; Wistuba et al., 2015; Olson et al., 2016a; Hutter et al., 2019). Among the tools of AutoML, the use of flat CV to both select the algorithms and hyperparameters is explicitly stated in Kotthoff et al. (2017) which describes Auto-WEKA, and we believe that other tools such as Auto-Sklearn (Feurer et al., 2015), TPOT (Olson et al., 2016b), and others also use flat CV. But, as far as the authors are aware, there has been no experimental or theoretical work to provide justification for that decision. The aim of this paper is to provide that experimental justification.

1.1. Estimating the Generalisation Performance of a Model

Let us denote by $ac(Y|a, X, \theta_a)$, the accuracy of algorithm a when trained on data X with hyperparameters θ_a and tested on data Y. Let us assume that a dataset G is an i.i.d. sample from some underlying distribution $\mathcal{D}$. The best algorithm for the dataset G is the algorithm that when trained on the whole dataset G, with the optimal values for the hyperparameters, will have the highest expected accuracy for future data.

The expected accuracy for future data (for algorithm a trained on G with hyperparameters θ_a) is:

$$\mathbb{E}_{g\sim\mathcal{D}}[ac(g|a, G, \theta_a)].$$

Given a set of candidate classification algorithms, $\mathcal{A}$, the best algorithm $\hat{a}$ is then:

$$\hat{a} = \underset{a\in\mathcal{A}}{\operatorname{argmax}} \ \mathbb{E}_{g\sim\mathcal{D}}[ac(g|a, G, \hat{\theta}_a)], \tag{1}$$

where $\hat{\theta}_a$ denote the best set of hyperparameters for algorithm a, that is:

$$\hat{\theta}_a = \underset{\theta_a}{\text{argmax}} \ \ \mathbb{E}_{g\sim\mathcal{D}}[ac(g|a, G, \theta_a)]. \tag{2}$$

Both nested and flat cross-validation procedures estimate the expected performance of the classifier, with optimal hyperparameter settings,

$$\mathbb{E}_{g\sim\mathcal{D}}[ac(g|a, G, \hat{\theta}_a)], \tag{3}$$

and then select the algorithm, $a \in \mathcal{A}$, having the highest estimate. Let us denote as $\widetilde{ac}^f(a, G)$ the flat cross-validation estimate of the term in Equation 3, and $\widetilde{ac}^n(a, G)$, the nested cross-validation estimate. Both estimates, $\widetilde{ac}^f(a, G)$ and $\widetilde{ac}^n(a, G)$, will likely result in different numeric values (Section 1.2).

The nested cross-validation (CV) procedure is considered more appropriate because $\widetilde{ac}^n(a, G)$ is an unbiased estimate of the expectation in Equation 3, whereas $\widetilde{ac}^f(a, G)$ has a positive bias (Cawley and Talbot, 2010), that is, on average, $\widetilde{ac}^f(a, G)$ will have higher, overly optimistic values than $\mathbb{E}_{g\sim\mathcal{D}}[ac(g|a, G, \hat{\theta}_a)]$. This arises as the data used in performance evaluation are also used indirectly in tuning the hyperparameters. Thus, the algorithm selected by the nested procedure is considered the more "correct" because the estimates of that procedure are unbiased relative to the true expected accuracy. However, the nested procedure has a much higher computational cost than the flat procedure. Therefore, one may want to use the flat procedure even at some risk of not selecting the best algorithm, if the computational expense is prohibitive. Notice that for the purpose of algorithm selection the positive bias of the flat procedure is not itself a problem provided the highest ranked algorithm by the nested procedure is the same as the highest ranked by the flat procedure, which implies the degree of bias is approximately the same for all classifiers.

Let us assume that $\hat{a}_n$ is the algorithm selected by the nested CV procedure, and $\hat{a}_f$ the algorithm selected using flat CV for dataset G (left implicit). Clearly, if $\hat{a}_n = \hat{a}_f$ in a very high percentage of cases, then one may choose the less expensive procedure, at some slight risk. In the cases where the two procedures do not agree on the best algorithm, we will compute the accuracy gain of the nested procedure selection relative to the flat selection procedure, or in other words, the difference in the expected accuracy on future data of the nested selection and the flat selection.

Let $\hat{\theta}_{\hat{a}_n}$ be the optimal hyperparameters for the $\hat{a}_n$ algorithm, and $\hat{\theta}_{\hat{a}_f}$ those of the $\hat{a}_f$ algorithm, as computed by the expression 2. Let us define the accuracy gain of using the nested CV procedure on dataset G as:

$$\text{accgain}(G) = \mathbb{E}_{g\sim\mathcal{D}}[ac(g|\hat{a}_n, G, \hat{\theta}_n)] - \mathbb{E}_{g\sim\mathcal{D}}[ac(g|\hat{a}_f, G, \hat{\theta}_f)], \tag{4}$$

where we left implicit the subscript $_G$ for the n and f symbols, for the sake of clarity. Of course, one cannot determine the true value of the accuracy gain, but as we will discuss in Section 2.1, we will be able to estimate it.

1.2. Flat and nested CV estimates

Both nested and flat CV procedures rely on using cross-validation to estimate an expectation of the form $\mathbb{E}_{g\sim\mathcal{D}}[ac(g|a,G,\theta_a)]$ when $\mathcal{D}$ is not known. Given a dataset G, cross-validation defines a set of pairs of sets TR_k and TE_k where TR_k is called a *training set*, and TE_k is called *test set* or sometimes *validation set*, and where:

$$TR_k \cap TE_k = \emptyset \quad \text{and} \quad TR_k \cup TE_k \subseteq G.$$

Common cross-validation procedures include k-fold, bootstrap, leave-one-out, hold-out, and so on.

Given a particular cross-validation procedure (which given G defines the sets TR_k and TE_k and the number of such pairs), the cross-validation estimate for the expected accuracy of the classifier (for a particular algorithm a and hyperparameters θ) is calculated as:

$$\text{mean}_k\ ac(TE_k|a, TR_k, \theta).$$

The flat CV estimate of $\mathbb{E}_g[ac(g|a,G,\hat{\theta}_a)]$ will select $\hat{\theta}_a^f$ as the value of θ that maximizes $\text{mean}_k ac(TE_k|a, TR_k, \theta)$:

$$\hat{\theta}_a^f = \underset{\theta}{\text{argmax}}\ \ \text{mean}_k\ ac(TE_k|a, TR_k, \theta), \tag{5}$$

and then use $\hat{\theta}_a^f$ to estimate $\mathbb{E}_g[ac(g|a,G,\hat{\theta}_a)]$, that is:

$$\widetilde{ac}^f(a,G) = \text{mean}_k\ ac(TE_k|a, TR_k, \hat{\theta}_a^f). \tag{6}$$

In nested CV, each training set (of the outer cross-validation) TR_k is further subdivided into pairs of sets of data TR_{km} and TE_{km} where again:

$$TR_{km} \cap TE_{km} = \emptyset \quad \text{and} \quad TR_{km} \cup TE_{km} \subseteq TR_k.$$

The nested cross-validation procedure will select the best hyperparameter $\hat{\theta}_a^k$ for each training set TR_k as:

$$\hat{\theta}_a^k = \underset{\theta}{\text{argmax}}\ \ \text{mean}_j\ ac(TE_{kj}|a, TR_{kj}, \theta).$$

The nested CV estimate of the expected accuracy for future data is:

$$\widetilde{ac}^n(a,G) = \text{mean}_k\ ac(TE_k|a, TR_k, \hat{\theta}_a^k).$$

Figure 1 gives an implementation of the flat CV as a Python program and Figure 2 provides the corresponding implementation for the nested CV procedure where the following functions are assumed:

- `createCV(data,...)` creates a list of pairs `(train,test)` from the `data`. Other parameters may include, for example, k if a k-fold CV procedure is used, or the proportion of cases in the training set, if a hold-out procedure is used.

```
def flat(data,...):
  cv = createCV(data, ...)
  acc_max = 0.0
  for theta in createGrid(...):
     acc = 0.0
     for train, test in cv:
        model = classtrain(train, theta)
        acc = acc + accuracy(model, test)
     if acc > acc_max:
        acc_max = acc
        theta_max = theta
  return acc_max / len(cv)
```

Figure 1: Implementation of the flat cross-validation procedure as a Python program.

- `createGrid()` creates the list of hyperparameter tuples to be tested. It could be a regular grid (in a grid search), or a random set of tuples (in a random search), or any other search algorithm.
- `classtrain(train,theta)` returns the classifier trained on data `train` with hyperparameters set to `theta`.
- `accuracy(model,test)` returns the accuracy (or any other quality measure) for the classifier `model` when run on data `test`.

The flat CV procedure computes and returns the final accuracy, as does the nested CV procedure. But the flat CV procedure also computes the best set of hyperparameters, and it could return it too. But, note that the nested CV procedure does not calculate a single best set of hyperparameter values; each training set of the outer cross-validation (k) may select different "optimal" hyperparameters.

When using a nested CV, the usual solution to determine a single best set of hyperparameter values is to use the flat cross-validation procedure to select them (Equation 5). Thus, when using a nested CV procedure to select among different algorithms, one would compute $\widetilde{ac}^n(a, G)$ for each algorithm a and select $\hat{a}$ with the highest estimate of the expected accuracy; for that algorithm, one would then perform the maximisation described in Equation 5 to select the best hyperparameters for $\hat{a}$. Essentially nested cross-validation estimates the performance of the full method used to generate the final model, including hyperparameter tuning.

The research presented here evaluated the mean accuracy gain of the nested CV procedure over flat-CV, by estimating its value over 115 real-life datasets, for 12 different classification algorithms. We show that the expected accuracy gain is very small, and we argue that the gain is of negligible practical consequence for most applications. That is, in the majority of cases, either the selection of the flat and nested procedures coincide, or the different best algorithms are so close in the expected accuracies that this difference can be considered irrelevant,

```
def nested(data,...):
  acc_final = 0.0
  cv_outer = createCV(data, ...)
  for tr_outer, te_outer in cv_outer:
     acc_max = 0.0
     for theta in createGrid(...):
       acc = 0.0
       cv_inner = createCV(tr_outer, ...)
       for tr_inner, te_inner in cv_inner:
          model = classtrain(tr_inner, theta)
          acc = acc + accuracy(model, te_inner)
       if acc > acc_max:
          acc_max = acc
          theta_max = theta
     model2 = classtrain(tr_outer, theta_max)
     acc_final = acc_final + accuracy(model2, te_outer)
  return acc_final / len(cv_outer)
```

Figure 2: Implementation of the nested cross-validation procedure as a Python program.

provided the algorithms have relatively few tuneable hyperparameters (as this strongly influences the bias of the flat-CV procedure).

2. Data and Methods

2.1. Experimental procedure

In this section, we set out in general terms the experimental procedure followed by this research. We performed 6 repetitions of a 50% split of each dataset into train and test subsets, each with the same proportion of patterns belonging to each class. For each dataset D_i, TR_r^i is the training subset for repetition r and TE_r^i is the corresponding test subset. For each train set, TR_r^i, we computed the expected accuracy using a 5-fold-within-5-fold nested-CV procedure ($\widetilde{ac}^n(a,i,r)$) and using a 5-fold flat-CV procedure ($\widetilde{ac}^f(a,i,r)$) for 12 different classification algorithms a (the classification algorithms are discussed in section 2.4). The flat-CV procedure also determines the best selection of hyperparameters ($\hat{\theta}_{air}$) for each algorithm a, for each TR_r^i. [1]

Let us define $\hat{a}_f(i,r)$ as the algorithm selected by the flat procedure on TR_r^i, and $\hat{a}_n(i,r)$ as the algorithm selected by the nested procedure. We define the **future accuracy** of an algorithm a on repetition r for dataset i as the accuracy of the algorithm when trained on TR_r^i with the best hyperparameters selected

[1] Following the nested cross-validation procedure, the selected model is re-trained on all of the available data, with 5-fold cross-validation based tuning of the hyperparameter values, which will, of course, give the same hyperparameter values as those already determined from the flat cross-validation trials.

by the flat procedure $\hat{\theta}_{air}$ and tested on TE_r^i. Formally the future accuracy on repetition r for dataset i is $ac(TE_r^i|a, TR_r^i, \hat{\theta}_{air})$, and intuitively, it is the accuracy for future data (TE_r^i) once the hyperparameters have been selected ($\hat{\theta}_{air}$), and the algorithm has been trained in the known data TR_r^i.

In particular we are interested in the future accuracy of the algorithms selected by the nested procedure $\hat{a}_n(i,r)$ and by the flat procedure $\hat{a}_f(i,r)$, and will define $\widetilde{fac}^n(i,r)$ as the future accuracy of the nested selection (for dataset i and round r) – similarly $\widetilde{fac}^f(i,r)$ is the future accuracy of the flat selection. Formally:

$$\begin{aligned}\widetilde{fac}^n(i,r) &= ac(TE_r^i|\hat{a}_n(i,r), TR_r^i, \hat{\theta}_{air}),\\ \widetilde{fac}^f(i,r) &= ac(TE_r^i|\hat{a}_f(i,r), TR_r^i, \hat{\theta}_{air}).\end{aligned} \tag{7}$$

The accuracy gain of using the nested procedure instead of the flat procedure is the difference between the future accuracy of the nested selection and the future accuracy of the flat selection,

$$accgain(i,r) = \widetilde{fac}^n(i,r) - \widetilde{fac}^f(i,r). \tag{8}$$

Finally, the accuracy gain of a dataset i is the average of the accuracy gains for the six rounds for that dataset:

$$accgain(i) = \frac{1}{6}\sum_{r=1}^{6} accgain(i,r). \tag{9}$$

Since the nested procedure is considered the "more correct" one, it should select the "more correct" algorithm, and thus it is more likely that the future accuracy of the nested selection would be higher than that of the flat selection. Thus, in general, one would expect a positive accuracy gain.

2.2. Threshold of irrelevance

To show that the least costly flat procedure achieves similar results (in future accuracy) as the nested procedure, we must show that the accuracy gains over all datasets are small. Unfortunately, there is no standard way of showing that an "aggregated" accuracy gain is small. A null hypothesis test will only determine if the aggregated accuracy gain is significantly different to 0; even if it is significantly different to 0 that difference may not be sufficiently large to be of *practical* significance. Also if the accuracy gain is not significantly different to 0, that does not establish that it actually is small, unless the statistical power of the test is high.

The more common approach to show that the difference between two sets of measurements is not of "practical significance" is to define a *single* threshold of irrelevance δ and then to show that it is likely that the differences are below this threshold. Within the frequentist approach, one must use *equivalence*

tests (Wellek, 2010) to show that it is unlikely that the differences are above the threshold. Within the Bayesian approach, the usual name for testing that differences are below the threshold is ROPE (region of practical equivalence). But both methods, Bayesian and frequentist, require a single threshold of irrelevance. For example, the threshold has been proposed at 1% (Benavoli et al., 2017) and 1.12% (Wainer, 2016), when comparing classifiers, and 0.3% when comparing resampling methods (Wainer and Cawley, 2017).

In this paper, we will follow a different approach, where the irrelevant threshold depends on the dataset. Each dataset i will have a irrelevance threshold $\delta(i)$, and we want to show that in general, or with high likelihood:

$$|accgain(i)| < \delta(i). \tag{10}$$

We use the Wilcoxon signed-rank test (a paired non-parametric test) to show that the median of the set $\{\ |accgain(i)|\ \}$ for each dataset i is smaller and significantly different than the median of the set $\{\ \delta(i)\ \}$. We also report the median and the 95% confidence interval of $\{|accgain(i)| - \delta(i)|$ for all $i\}$ so the reader may gain a sense of the magnitude of the differences. The confidence interval was calculated using bootstrap with 5000 rounds.

Our idea for a threshold of irrelevance is based on unavoidable errors in the accuracy estimate; unavoidable because they depend on random factors, such as the sampling of the data to form training and test sets. The threshold depends both on the dataset and the algorithm. If the dataset is small one expects larger changes in accuracy when different splits of train and test or when comparing estimated accuracy with the real accuracy on future, unseen data. If the algorithm overfits the data, or if the algorithm underfits the data, one would also expect larger differences in the accuracy in those different conditions.

Our proposal for the irrelevance threshold δ is based on the idea that the nested procedure estimate of the future accuracy is only an estimate of the actual generalisation performance. Differences between the estimate and the measured accuracy for some unseen data may indicate how sensitive is the combination of dataset and algorithm to these unavoidable variations. We define $\Delta(a, i, r)$ as the difference between the nested estimate of future accuracy and the measured future accuracy for a particular algorithm a, dataset i, and repetition r. Formally:

$$\Delta(a, i, r) = |\widetilde{ac}^n(a, i, r) - \widetilde{fac}(a, i, r)|. \tag{11}$$

The threshold of irrelevance for a dataset i and round r, $\delta(i, r)$, is the minimum between $\Delta(\hat{a}_n(i, r), i, r)$ and $\Delta(\hat{a}_f(i, r), i, r)$

$$\delta(i, r) = \min \Delta(\hat{a}_n(i, r), i, r), \Delta(\hat{a}_f(i, r), i, r), \tag{12}$$

The idea is that the threshold of irrelevance for a dataset and a round is the smallest of the errors between estimated and measured future accuracy for the two "important/best" algorithms for that dataset and for that round, $\hat{a}_n(i, r)$ and $\hat{a}_f(i, r)$. The reason to take the minimum is to achieve a more restrictive definition of irrelevance.

The final threshold for the dataset $\delta(i)$ is the average of $\delta(i, r)$ for all repetitions:

$$\delta(i) = \frac{1}{6} \sum_{r} \delta(i, r) \tag{13}$$

Appendix A discusses another definition for the irrelevance threshold and the analysis of the results using that definition.

Finally, it is interesting to understand the role of the repetition in this experimental procedure. Repetitions r are seen as different experiments to compute the accuracy gain of the nested procedure versus the flat procedure. Each repetition may select different algorithms in the nested and in the flat procedures. The goal of the experiment/repetition is to compute the accuracy gain (Equation 8) and the irrelevance threshold (Equation 12). Only then are the accuracy gain and irrelevance thresholds aggregated across repetitions on the same dataset (Equations 9 and 13).

This form of analysis is inspired by the nested cross-validation procedure, which only aggregates the data on the different folds/hold-out subsets to compute the final measure of interest, the expected accuracy. The two measures of interest in this research are the accuracy gain and the threshold of irrelevance, and only at that level, the results are averaged across repetitions. Appendix B discusses different ways of using the repetitions and presents the corresponding results.

2.3. Scenarios

In this paper, we are interested in answering two questions regarding the nested and flat selection procedures. The first question is whether one need use a nested procedure to select the best among three very good algorithms for classification: random forest (rf), SVM with RBF kernel (svmRadial), and gradient boosting machine (gbm). There is some independent evidence to suggest that these three algorithms are likely the best classification algorithms in general. Fernández-Delgado et al. (2014) do not test gradient boosting machines, and find that random forest and SVM with RBF kernel are the two best families of algorithms. Wainer (2016) does test gradient boosting machines, and finds that those three form the best three families of classification algorithms.

We do not make the claim that rf, svmRadial, and gbm *are*, in general, the best algorithms to binary classification problems. There are some literature that claims that other algorithms may perform as well or better than those three, for example (Bagnall et al., 2018; Cañete-Sifuentes et al., 2019). We use rf, svmRadial, and gbm as example of well performing algorithms that has been compared and tested by multiple authors, and that have multiple, well established implementations.

As we will discuss in section 3, this research does find that random forest is the algorithm with lowest mean rank, followed by SVM with RBF, followed by gradient boosting machines. Thus, practitioners that have a restriction on the amount of time needed to select the best classification algorithm should

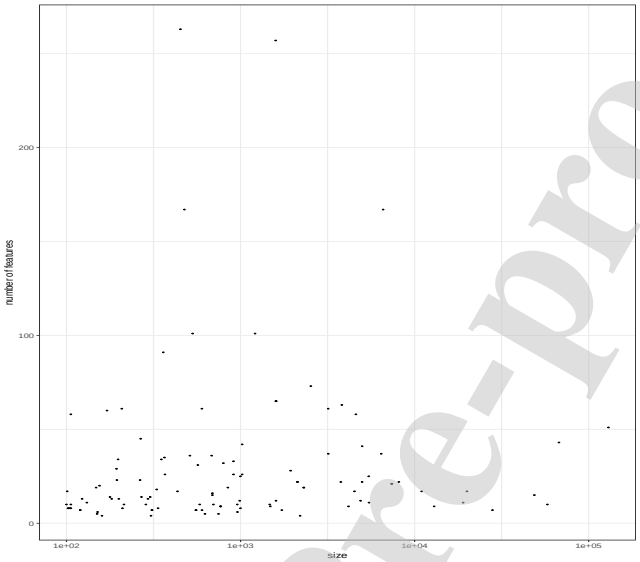

Figure 3: The joint distribution of datasets sizes and number of features

restrict themselves to these three algorithms. The first question we will address is whether, when selecting among rf, svmRadial, and gbm, one can avoid the nested procedure and use a flat procedure instead. In this scenario, called **top3**, we restrict the analysis to only those three algorithms.

The second question is whether the nested procedure is necessary when any set of classifiers are being compared. In this case, we tested 12 different families of classifiers (the algorithms are discussed in Section 2.4). We call this the full scenario.

2.4. Datasets and classification algorithms

We used the suite of datasets collected from the UCI public repository and processed by Fernández-Delgado et al. (2014) and further processed by Wainer (2016), such that all datasets are binary classification tasks. Figure 3 plots the joint distribution of dataset's sizes and number of features/dimensions. For the 9 datasets with more than 10000 data points, we applied the procedures (nested and flat CV) on only a random subset of 5000 data points (from each subset).

For each subset, we applied 12 different classification algorithms. The algorithms and their abbreviations are as follows:

bst A boosting of linear classifiers. The hyperparameters searched for this classifiers were: shrinkage and the number of boosts.

gbm Gradient boosting machines — a boosting of short decision trees (Friedman, 2001). Hyperparameters: interaction depth, shrinkage, and the number of boosts.

knn The k-nearest neighbours classifier (Dasarathy, 1991). Hyperparameter: k.

lvq Learning vector quantisation (Kohonen, 1995). Hyperparameter: the size of the codebook.

nb Naïve Bayes classifier (Hand and Yu, 2001). Hyperparameters: Laplace smoothing constant.

nnet A 1-hidden layer neural network with sigmoid transfer function (Bishop, 1995).Hyperparameters: number of hidden units and decay.

rf Random forest — bagging of decision trees (Ho, 1998). Hyperparameters: number of trees and number of features to search in each split.

rknn A bagging of k-nn classifiers on a random subset of the original features (Dasarathy, 1991). Hyperparameters: k, number of k-nn classifiers, and number of dimensions in each k-nn.

sda A L1 regularised linear discriminant classifier (Ahdesmäki et al., 2010). Hyperparameter: the regularisation constant λ

svmLinear A SVM with linear kernel (Cortes and Vapnik, 1995). Hyperparameter: C

svmPoly A SVM with polynomial kernel (Cortes and Vapnik, 1995). Hyperparameters: C and the degree.

svmRadial A SVM with RBF kernel (Cortes and Vapnik, 1995). Hyperparameters: C andγ.

Details of the particular implementations of these algorithms and hyperparameters ranges and search grids are described in Wainer (2016). But notice that all algorithms have a low number of hyperparameters to be searched, varying from 1 (knn, nb, lvq, sda, and svmLinear) to 3 (gbm and rknn).

2.5. Reproducibility

The datasets, the program to run the different procedures and the different classifiers, the results of the multiple runs, and the R program to perform the statistical analysis described in this paper are available at `https://doi.org/10.6084/m9.figshare.3457238` .

3. Results

Table 1 lists the mean rankings of the algorithms, according to the nested CV estimate of their accuracies, over all repetitions and over all datasets.
The results of the top-3 agree with the order in Wainer (2016).

algorithm	mean rank
rf	3.4
svmRadial	3.6
gbm	4.0
nnet	4.8
rknn	5.3
svmPoly	5.3
knn	5.4
svmLinear	6.1
sda	6.6
lvq	6.7
nb	7.9
bst	8.7

Table 1: Ranking of the algorithms based on the mean rank for each repetition.

3.1. Results for the top-3 and full scenarios

Figure 4 displays the accuracy gain and the thresholds of irrelevance for the top-3 scenario (random forest, SVM with RBF kernel, and gradient boosting machines). The figure relates each measure of the $\log_{10}$ of the accuracy gain (in the vertical) with the corresponding $\log_{10}$ of the threshold of irrelevance (horizontal). The triangle points are data points with 0 accuracy gain, which were artificially placed in the $\log_{10}$ accuracy gain $= -5$ line. Notice that most points are in the lower part of the $y = x$ line, which shows that in most cases, the threshold of irrelevance is higher than the corresponding accuracy gain. Figure 5 displays the corresponding comparison of the accuracy gain and threshold of irrelevance for the full scenario.

Table 2 displays the results for statistical analysis for the top-3 and full scenarios. "Same choice" is the proportion of times the algorithm selected using flat CV agreed with that selected using nested CV. The column "p.value" is the p-value of the one-sided Wilcoxon signed-rank test between the accuracy gain and the irrelevance threshold. The "median" column is the median of the difference of the accuracy gain and the irrelevance threshold, and it is negative as expected, the "low CI" and "high CI" columns are the lower and higher limits of the 95% confidence interval for the median.

For the top-3 scenario, the flat procedure selects the same algorithm that the nested procedure selects in 71% of the cases (a random choice would give a figure of 33%). The p-value is below 0.05, which shows that the accuracy gains for the nested procedure are significantly smaller than the corresponding thresholds of irrelevance. Therefore, one can claim that the accuracy gain is statistically significantly less than the corresponding irrelevance threshold (at the 95% level of significance). Thus our claim that there is no practical difference on average, between using either the nested or the flat procedure to select among random forest, SVM with RBF kernel, and gradient boosting machines. For the full scenario, the agreement rate between flat and nested is 62% (against 8% if the

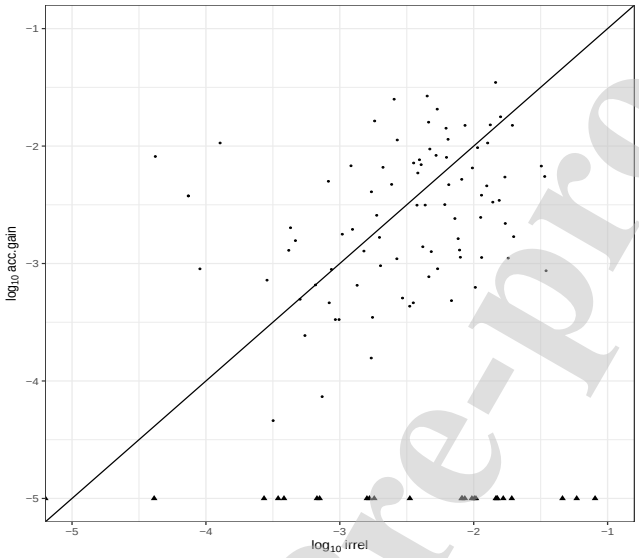

Figure 4: Comparison of $\log_{10}$ accuracy gain and $\log_{10}$ irrelevance threshold for the top-3 scenario. The triangular data points represent the data where the accuracy gain is zero.

decision was random), and again a p-value below 0.05. Therefore, again, one can be confident that on average the accuracy gain is below the corresponding threshold of irrelevance.

scenario	same choice	p.value	median	low CI	high CI
top-3	71%	0.001	-0.001	-0.002	0.0
full	62%	3.6e-06	-0.001	-0.002	-0.001

Table 2: The results for the selection of the top-3 and full scenarios. The column "Same choice" is the proportion of times the selection using flat CV agreed with the selection using nested CV. "p.value" is the p-value of the one-sided Wilcoxon signed rank test of the accuracy gain and the corresponding threshold. "median" is the median value of the difference $|accgain(i)|-\delta(i)$, and "low CI" and "high CI" are the limits of the 95% confidence interval of that median.

3.2. Results for other metrics

The paper so far has discussed that there is no need for nested cross validation when selecting classifier algorithms, for both the top-3 and full scenarios, when using accuracy as metric of quality. But accuracy it is not the only metric used to evaluate the quality of a classifier, specially for 2-class problems. Area under the ROC curve (AUC) and f1 measure are very frequently used quality metrics for binary problems, specially if the two classes are not balanced. In this

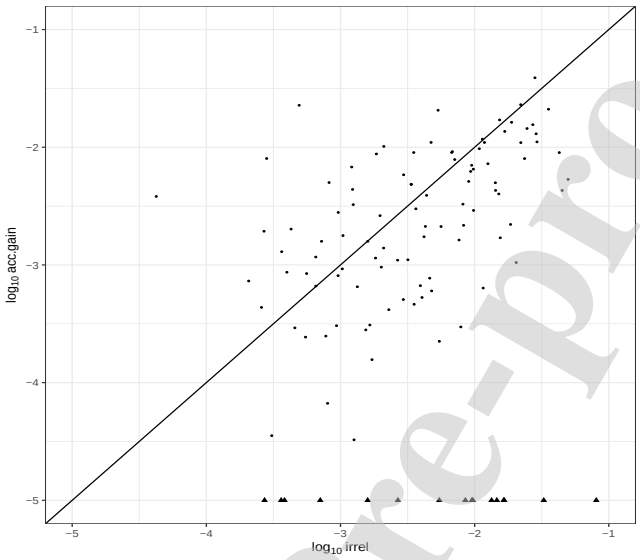

Figure 5: Comparison of $\log_{10}$ accuracy gain and $\log_{10}$ irrelevance threshold for the full scenario. The triangular data points represent the data where the accuracy gain is zero.

section we report the results of performing the experiments above, in the top-3 scenario, but using AUC and f1-measure to select the best hyperparameters and algorithms. The results are displayed in Table 3.

metric	same choice	p.value	median	low CI	high CI
AUC	76%	0.0001	-0.001	-0.001	0.0
F1	78%	0.0001	-0.002	-0.006	-0.001

Table 3: The results for the selection of the top-3 scenario, using AUC and F1 as the quality metrics.

The result are very similar to the top-3 line in Table 2. Using these two metrics to select among rf, svm, and gbm, one should expect over 75% of agreement between the selection using flat and nested cross validation. Furthermore, with 95% confidence, the gains of the nested procedure when the selections do not agree is below the threshold of irrelevance, which suggests that our conclusions regarding the practical need for nested cross validation is independent of the metric used.

4. Discussion

This paper makes two claims:

- Nested CV procedures are probably not needed when selecting from among random forest, SVM with Gaussian kernel, and gradient boosting machines (which are on average the three best classification algorithms for the suite of datasets used in this research), when using accuracy, AUC, or F1-measure as quality metrics.

- Nested CV procedures are probably not needed when selecting from among any set of classifier algorithms (provided they have only a limited number of hyper-parameters that must be tuned as we will discuss below), when using accuracy, AUC, or F1-measure as quality metrics.

The first claim was explicitly tested on 115 datasets and thus, to generalize it the reader must believe that the 115 datasets are an unbiased sample of datasets a practitioner will face in the future. We discuss the limits of such generalisation below. This second claim carries others risk for generalisation, namely whether the full set of 12 classifiers is a good sample of the sets of classifiers that will be selected in future applications, and whether our tests on the top-3 scenario with other three metrics can be further generalised to the full scenario.

Some of the limits to the generalisation of conclusions obtained from the set of 115 datasets to any future dataset are discussed in Wainer (2016). Briefly, the datasets tested in this research were only of medium size (up to 100,000 data points), only binary datasets were used, and none of them is derived from text classification problems (with high dimensionality and high sparsity). It is not immediately obvious how the number of dimensions, sparsity, or the fact that there are more than two classes could have a substantial impact on the claims made in this research. Dataset size could be an issue as the bias introduced by the flat cross-validation procedure generally decreases as the size of the dataset increases. Thus, if nested cross-validation is not generally necessary for small or medium-sized datasets, it is even less likely to be necessary for large datasets. The limit of irrelevance is the difference between the nested estimate of accuracy for future data and a measured accuracy on future data, and this difference should also decrease as the dataset sizes increase. In other words, the variance of the nested estimate should decrease as the dataset size increases, and so should the variance of the hold-out subset that we used to measure the "true" accuracy for future data, and thus the difference should decrease. We measured whether our claim of practical equivalence between the two procedures had a dependency on the dataset size. Table 4 reports the statistical tests for the two scenarios, only for the 32 datasets with 2000 data or more. For the larger datasets only, the strength of the evidence in favour of the practical equivalence of the nested and flat procedure diminishes, as expected, given that there are fewer datasets/measures used in the significance test, but one can still make the claim of the practical equivalence of the nested and flat procedures.

The second claim, that for all classification algorithms (with small number of hyperparameters) nested cross validation is probably do not needed to select the hyperparameters require a double generalisation. As before, one must accept that the 115 datasets we tested this claim is a "representative" sample of

scenario	same choice	p.value	median	low CI	high CI
top-3	80%	0.00158	-0.001	-0.002	0.0
full	71%	0.0108	0.0	-0.002	0.0

Table 4: The results for 32 datasets with at least 2000 data points.

datasets the practitioner will face in the future, and the limits of this generalisation has been discussed above. The second generalisation is that out teste on 12 classification algorithms is a "representative" sample of classification algorithms the practitioner will use in the future.

Our choice of classification algorithms was in support of two goals:

- test algorithms that would be likely used by a practitioner in a future classification task. Therefore we included algorithms like SVM with RBF kernels, random forests, and gradient boosting machines, which have been shown by different groups using different tests and methodology to be among the algorithms that result in higher accuracy for most of the problems. We are aware that there are other algorithms that may be competitive with these three, for example, rotation forests (Bagnall et al., 2018), Gaussian process based classifiers (for example (Gibbs and MacKay, 2000)), and others (Cañete-Sifuentes et al., 2019). We do not claim that the algorithm in the top 3 scenario *are* the likely best algorithms, but that they are widely believe to be among the best algorithms for classification, they are well established and have stable and maintained implementations, and are likely the first choices a practitioner will use on their data.

- most of the remaning algorithms are algorithms that are less likely to be the best solutions for a particular problem. Nonetheless there are also computational reasons to choose some of the algorithms among this second group. But the main rational of the second group was to select from a wide variety of "families" of algorithms - algorithms that are based on very different principles, as an attempt to select a "representative" sample of future classification algorithms. The inclusion of less known "families" of algorithms such as random subspace knn (rknn), regularised discriminant analysis (sda), and learning vector quantisation (lvq) was exactly to have different families of algorithms, instead of variations within a single family (for example different variations of boosting (Schapire and Freund, 2013)). We are aware that there are other families of algorithms that were not tested, for example regularised logistic regression (Zou and Hastie, 2005) Gaussian process classifiers (Gibbs and MacKay, 2000), rule induction algorithms (Cohen, 1995), evolutionary fuzzy rule induction (Del Jesus et al., 2004), extreme learning machines (Huang et al., 2011). The decision to leave these and many other "families" of classifiers out of the test was necessarily ad hoc.

The strength of the paper's second claim that one does not need nested-cv for any classification algorithm rest on the reader's acceptance that our choice

of algorithms tested is a "representative" sample of classification algorithms.

All algorithms tested in this research had a small number of hyperparameters, from 1 to at most 3. Algorithms with many more hyperparameters will likely pose a challenge to the conclusion that flat CV is sufficient to select the algorithms. An interesting example is discussed in Cawley and Talbot (2010). An ARD kernel has a different γ for each original data dimension. In principle, an LS-SVM with ARD subsumes the standard RBF LS-SVM, and thus it should not have an expected accuracy lower than the classical RBF LS-SVM. But Cawley and Talbot (2010) shows that although the LS-SVM with ARD kernel achieves a higher expected accuracy when using the flat CV estimate of the accuracy, when using the nested procedure, the classical RBF LS-SVM has a statistically significant higher accuracy in 7 out of 13 datasets tested, while the ARC LS-SVM is statistically better in only one of those 13 datasets (Cawley and Talbot, 2010, Table 2). In this case, because the ARD LS-SVM has so many more hyperparameters than the RBF LS-SVM, the flat procedure will likely overfit the data. Thus, in the case of algorithms with very different number of hyperparameters (such as ARD based algorithms, multiple hidden layer neural networks, and deep networks), we feel less confidence in our practical equivalence results between the nested and flat procedures.

Finally, we only tested the use of other quality metrics for the top-3 scenario, but given that the results for that scenario are very close among accuracy, AUC, and F1 we are very confident that the results for accuracy that were measure would repeat for the other two metrics in the full scenario.

Appendix B shows that the conclusions reached by this paper do not strongly depend on the method of analysis – two other methods of analysis result in the same conclusions. Appendix B also shows that the results remain even when a different definition of the threshold of irrelevance is used. Appendix C shows that one should not go a step further and skip the selection of the algorithm altogether – in this case mean accuracy gain is significantly larger than the threshold of irrelevance.

The results of this paper only partially support the current practice in AutoML research of using flat cross-validation to select both classifier and its hyperparameters. Using AutoML procedures to select among algorithms with similar number of hyperparameters is justified by this research. But a particular popular algorithm XGBoost (Chen and Guestrin, 2016), which by changing hyperparameters can implement from gradient boosting machines to random forests, has many more hyperparameters than more traditional classification algorithms. As we discussed above, in cases of very different number of hyperparameters, we are less confident of the equivalence of flat and nested procedures.

The results in this paper are only applicable to practitioners, that is, for users that have the goal of selecting the likely best classification algorithm to solve a particular problem. Our results cannot be applied by a scientist whose goal is to provide evidence that one classification algorithm is better than another. Our claim of practical equivalence applies only to the best-ranked algorithm for both procedures, and not that the two procedures have some significant agreement regarding the full ranking. For example, Table 5 list the rank of

the 12 algorithms when using the flat procedure estimate to order them. The table should be compared to Table 1. The order of the algorithms is very different; in particular, using the flat estimate, the gbm would be classified as the best algorithm while using the nested CV estimate, it is ranked third. In particular, given that the gbm has 1 hyperparameter more than svmRadial or rf, we believe that this improvement in the ranking could be due to the model overfitting described above (Cawley and Talbot, 2010).

algorithm	mean flat-CV rank
gbm	3.0
svmRadial	3.2
rf	4.0
nnet	4.1
rknn	4.2
svmPoly	5.2
knn	5.3
lvq	6.4
svmLinear	6.4
sda	7.0
nb	8.4
bst	8.6

Table 5: Ranking of the algorithms based on the mean rank for each subset ordered by the flat CV estimate of the expected error.

The distribution of the values of $\delta(i)$ is important for the machine learning community that may be interested in determining when differences are of no practical significance, using either method. The summary of the distribution is shown in Table 6 and the histogram in Figure 6. The median threshold of irrelevance is 0.4% and the mean 0.9%, not too different, but lower than the values proposed in the previous literature (Benavoli et al., 2017; Wainer, 2016).

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00%	0.11%	0.41%	0.89%	1.17%	8.06%

Table 6: Summary of the distribution of irrelevance thresholds.

5. Conclusion

There is very strong evidence that when selecting among a random forest, an SVM with Gaussian kernel, and a gradient boosting machine (the three best algorithms on average for the 115 real-life datasets tested) one can generally use the flat cross-validation procedure to search for the best hyperparameters **and** to select the best algorithm itself. Our analysis shows that the algorithm selected by the flat procedure will, on average, perform as well as the one that would be selected by the nested cross-validation procedure, for most practical

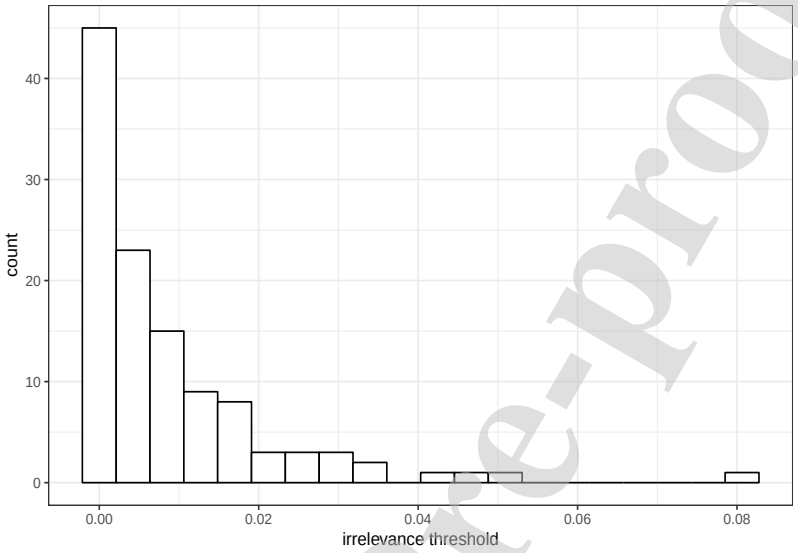

Figure 6: Histogram of the values of the irrelevance thresholds $\delta(i)$

purposes. Also, there is some indication that the conclusions remain even for datasets larger than the ones tested.

There is also a strong evidence that in any selection process, regardless of the algorithms that are being selected, provided they all have a low number of hyperparameters, one can use the flat cross-validation procedure to select the algorithm and the hyperparameters simultaneously, and again for all practical purposes, that algorithm would perform as well as the algorithm selected using nested cross-validation.

Acknowledgements

We would like to thank Nicola Talbot for important contributions to the paper. We would also like to thank an anonymous reviewer for suggesting the AUC and F1 experiments.

References

References

Ahdesmäki, M., Strimmer, K., et al., 2010. Feature selection in omics prediction problems using CAT scores and false nondiscovery rate control. The Annals of Applied Statistics 4, 503–519.

Bagnall, A., Flynn, M., Large, J., Line, J., Bostrom, A., Cawley, G., 2018. Is rotation forest the best classifier for problems with continuous features? arXiv preprint arXiv:1809.06705 .

Benavoli, A., Corani, G., Demšar, J., Zaffalon, M., 2017. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. The Journal of Machine Learning Research 18, 2653–2688.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.

Cañete-Sifuentes, L., Monroy, R., Medina-Pérez, M.A., Loyola-González, O., Voronisky, F.V., 2019. Classification based on multivariate contrast patterns. IEEE Access 7, 55744–55762.

Cawley, G., Talbot, N., 2010. On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research 11, 2079–2107.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 785–794.

Cohen, W.W., 1995. Fast effective rule induction, in: Machine learning proceedings 1995. Elsevier, pp. 115–123.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20, 273–297.

Dasarathy, B.V. (Ed.), 1991. Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press.

Del Jesus, M.J., Hoffmann, F., Navascués, L.J., Sánchez, L., 2004. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Transactions on Fuzzy Systems 12, 296–308.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research 15, 3133–3181.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F., 2015. Efficient and robust automated machine learning, in: Advances in neural information processing systems, pp. 2962–2970.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.

Gibbs, M.N., MacKay, D.J., 2000. Variational gaussian process classifiers. IEEE Transactions on Neural Networks 11, 1458–1464.

Hand, D.J., Yu, K., 2001. Idiot's Bayes — not so stupid after all? International Statistical Review 69, 385–398.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 832–844.

Huang, G.B., Wang, D.H., Lan, Y., 2011. Extreme learning machines: a survey. International journal of machine learning and cybernetics 2, 107–122.

Hutter, F., Kotthoff, L., Vanschoren, J., 2019. Automatic Machine Learning: Methods, Systems, Challenges. Springer.

Kohonen, T., 1995. Learning vector quantization. Springer.

Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K., 2017. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. The Journal of Machine Learning Research 18, 826–830.

Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H., 2016a. Evaluation of a tree-based pipeline optimization tool for automating data science, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM. pp. 485–492.

Olson, R.S., Urbanowicz, R.J., Andrews, P.C., Lavender, N.A., Moore, J.H., et al., 2016b. Automating biomedical data science through tree-based pipeline optimization, in: European Conference on the Applications of Evolutionary Computation, Springer. pp. 123–137.

Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: A new classifier ensemble method. IEEE transactions on pattern analysis and machine intelligence 28, 1619–1630.

Schapire, R.E., Freund, Y., 2013. Boosting: Foundations and algorithms. Kybernetes .

Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society. Series B (Methodological) , 111–147.

Wainer, J., 2016. Comparison of 14 different families of classification algorithms on 115 binary datasets. `arXiv:arXiv:1606.00930`.

Wainer, J., Cawley, G., 2017. Empirical evaluation of resampling procedures for optimising SVM hyperparameters. The Journal of Machine Learning Research 18, 475–509.

Wellek, S., 2010. Testing statistical hypotheses of equivalence and noninferiority. Chapman and Hall/CRC.

Wistuba, M., Schilling, N., Schmidt-Thieme, L., 2015. Sequential model-free hyperparameter tuning, in: 2015 IEEE international conference on data mining, IEEE. pp. 1033–1038.

Wolpert, D.H., 1996. The lack of a priori distinctions between learning algorithms 8, 1341–1390.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology) 67, 301–320.

Appendix A. Another definition of irrelevance

This appendix discusses another measure that can play the role of irrelevance threshold and the analysis of the data when using this new definition. When we discussed the irrelevance threshold, we mentioned unavoidable error or unavoidable variance, and we chose the mean difference between the nested estimate of accuracy and the true measure of accuracy on the test set. But standard deviation is a common way of measuring error that we could use instead of the mean difference of two accuracies as we did. There are three measures of accuracy for each repetition: $\widetilde{ac}^n(a,i,r)$, $\widetilde{ac}^f(a,i,r)$, and $ac(TE^i_r|a, TR^i_r, \hat{\theta}_{ai})$. We define the threshold for each dataset as

$$\delta(a,i) = \min(\sigma_r(\widetilde{ac}^n(a,i,r)), \sigma_r(\widetilde{ac}^f(a,i,r)), \sigma_r(ac(TE^i_r|a, TR^i_r, \hat{\theta}_{ai}))$$

that is, the smallest of the three standard deviations of measured accuracies across the six repetitions.

Using this definition of the threshold of irrelevance (and using the paper's original method of analysis) results in:

scenario	same choice	p.value	median	low CI	high CI
top-3	80%	5.94e-06	-0.002	-0.003	-0.001
full	71%	1e-06	-0.001	-0.002	-0.001

These results are equivalent to the ones discussed in the paper.

Appendix B. Other analysis methods

As discussed, the analysis method in this paper assumes that each repetition is an independent experiment, and the repetitions are only aggregated at the last step, to compute the accuracy gain and the threshold of irrelevance for a dataset. But there are some alternatives to that analysis method. The first alternative is to consider the repetition as a way of obtaining multiple estimates

for each of the accuracies measures. Thus, all measured accuracies are first averaged across the six repetitions and only then used in the procedure, that is:

$$\widetilde{ac}^n(a,i) = \frac{1}{6}\sum_r \widetilde{ac}^n(a,i,r)$$
$$\widetilde{ac}^f(a,i) = \frac{1}{6}\sum_r \widetilde{ac}^f(a,i,r)$$

The flat and nested selections for each dataset ($\hat{a}_f(i)$ and $\hat{a}_n(i)$) would be selected using $\widetilde{ac}^n(a,i)$ and $\widetilde{ac}^f(a,i)$ (in contrast to the method used which selects $\hat{a}_f(i,r)$ and $\hat{a}_n(i,r)$ for each repetition). Then equations 7 and 8 would be

$$\widetilde{fac}(n,i) = \frac{1}{6}\sum_r ac(TE_r^i|\hat{a}_n(i), TR_r^i, \hat{\theta}_{ni}).$$
$$\widetilde{fac}(f,i) = \frac{1}{6}\sum_r ac(TE_r^i|\hat{a}_f(i), TR_r^i, \hat{\theta}_{fi}).$$

and

$$accgain(i) = \widetilde{fac}(n,i) - \widetilde{fac}(f,i)$$

Similarly, the thresholds of irrelevance are not defined for each repetition but only for each dataset:

$$\delta(a,i) = |\widetilde{ac}^n(a,i) - \widetilde{fac}(a,i)|$$

The second alternative is to consider each repetition as an independent experiment at par with the dataset themselves. The results for each dataset it only aggregated at the last level, when considering the p-value of the Wilcoxon test that compares $|accgain(i)| - \delta(i)$ with 0. In this second alternative, we would perform the Wilcoxon test to compare $|accgain(i,r)| - \delta(i,r)$ to 0.
The first alternative method above yields the following results:

scenario	same choice	p.value	median	low CI	high CI
top-3	78%	6.32e-06	-0.002	-0.003	-0.001
full	72%	0.0005	-0.001	-0.003	0.0

The second method yields the following results:

scenario	same choice	p.value	median	low CI	high CI
top-3	78%	3.05e-13	-0.001	-0.001	-0.001
full	72%	2.67e-08	-0.001	-0.001	0.0

These different methods of analysis are consistent with the claims of this paper.

Appendix C. Should one select the algorithm at all?

Given that our research shows an unexpected result that flat CV is acceptable as a method to select classification algorithms, contrary to the common

practice in Machine Learning, we decided to explore another unexpected result, whether the selection of algorithms is really necessary, or if one should just use random forests, which was the best-ranked algorithm in the experiments. We compared the decision of using only rf against the nested procedure. The results are below:

scenario	same choice	p.value	mean	low CI	high CI
full	28%	1.0	0.002	0	0.004

The results show that the accuracy gain is certainly above the threshold of irrelevance, and thus selecting the algorithm results in an expected accuracy gain of practical consequence.

Highlights (for review)

- flat cross validation computes both the best hyperparameter and the expected accuracy
- nested cross validation separated both computations and it is more costly
- nested cross validation does not incur on biased estimation of the accuracy
- algorithm selection using flat cross validation does not incur in worse selections

ORCID Information

Jacques Wainer **ORCID 0000-0001-5201-1244**

Gavin Cawley **no ORCID ID**

Credit Author Statement

JW and GC proposed the methodology. JW ran the experiments, JW wrote the first draft. JW and GC wrote the final version of the paper.

Declaration of Interest Statement

December 20, 2019

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Jacques Wainer and Gavin Cawley