International Journal on Artificial Intelligence Tools
Vol. 24, No. 5 (2015) 1540023 (29 pages)

DOI: 10.1142/S0218213015400230

World Scientific
www.worldscientific.com

# Performance-Estimation Properties of Cross-Validation-Based Protocols with Simultaneous Hyper-Parameter Optimization

Ioannis Tsamardinos* and Amin Rakhshani†

*Department of Computer Science, University of Crete, Crete, Greece*
*and*
*Institute of Computer Science, Foundation for Research and Technology — Hellas (FORTH)*
*Heraklion Campus, Voutes, Heraklion, GR-700 13, Greece*
**tsamard.it@gmail.com*
†*aminra@ics.forth.gr*

Vincenzo Lagani‡

*Institute of Computer Science, Foundation for Research and Technology — Hellas (FORTH)*
*Vassilika Vouton, Heraklion, GR–700 13, Greece*
*vlagani@ics.forth.gr*

Received 12 October 2014
Accepted 23 June 2015
Published 20 October 2015

In a typical supervised data analysis task, one needs to perform the following two tasks: (a) select an optimal combination of learning methods (e.g., for variable selection and classifier) and tune their hyper-parameters (e.g., K in K-NN), also called *model selection*, and (b) provide an estimate of the performance of the final, reported model. Combining the two tasks is not trivial because when one selects the set of hyper-parameters that seem to provide the best estimated performance, this estimation is optimistic (biased/overfitted) due to performing multiple statistical comparisons. In this paper, we discuss the theoretical properties of performance estimation when model selection is present and we confirm that the simple Cross-Validation with model selection is indeed optimistic (overestimates performance) in small sample scenarios and should be avoided. We present in detail and investigate the theoretical properties of the Nested Cross Validation and a method by Tibshirani and Tibshirani for removing the estimation bias. In computational experiments with real datasets both protocols provide conservative estimation of performance and should be preferred. These statements hold true even if feature selection is performed as preprocessing.

*Keywords*: Performance estimation; model selection; cross validation; stratification; comparative evaluation.

## 1. Introduction

A typical supervised analysis (e.g., classification or regression) consists of several steps that result in a final, single prediction, or diagnostic model. For example, the analyst may

‡Corresponding author.

need to impute missing values, perform variable selection or general dimensionality reduction, discretize variables, try several different representations of the data, and finally, apply a learning algorithm for classification or regression. Each of these steps requires a selection of algorithms out of hundreds or even thousands of possible choices, as well as the tuning of their hyper-parameters.[a] *Hyper-parameter optimization* is also called the *model selection* problem since each combination of hyper-parameters tried leads to a possible classification or regression model out of which the best is to be selected. There are several alternatives in the literature about how to identify a good combination of methods and their hyper-parameters (e.g., Refs. 1 and 2) and they all involve implicitly or explicitly searching the space of hyper-parameters and trying different combinations. Unfortunately, trying multiple combinations, estimating their performance, and *reporting the performance of the best model found leads to overestimating* the performance (i.e., *underestimate the error/loss*), sometimes also referred to as overfitting.[b] This phenomenon is called the problem of *multiple comparisons in induction algorithms* and has been analyzed in detail in Ref. 3 and is related to the *multiple testing* or *multiple comparisons* in statistical hypothesis testing. Intuitively, when one selects among several models whose estimations vary around their true mean value, it becomes likely that what seems to be the best model has been "lucky" in the specific test set and its performance has been overestimated. Extensive discussions and experiments on the subject can be found in Ref. 2.

An intuitive small example now follows. Let us suppose method $M_1$ has 85% true accuracy and method $M_2$ has 83% true accuracy on a given classification task when trained with a randomly selected dataset of a given size. In 4 randomly drawn training and corresponding test sets on the same problem, the estimations of accuracy maybe 80, 82, 88, 90 for $M_1$ and 88, 85, 79, 79 percent. If $M_1$ was evaluated by itself the estimated mean accuracy will be estimated as 85%, and for $M_2$ it would be 82,75% respectively, that are close to their true means. If performance estimations were perfect then $M_1$ would be chosen *each time* and the average performance of the models returned *with model selection* would be 85%. However, when both methods are tried, the best is selected, and the maximum performance is reported, we obtain the series of estimations: 88, 85, 88, 90 whose average is 87, 75 and will be in generally biased. A larger example and contrived experiment now follows:

**Example.** In a binary classification problem, an analyst tries $N$ different classification algorithms, producing $N$ corresponding models from the data. They estimate the performance (accuracy) of each model on a test set of $M$ samples. They then select the model

[a]We use the term "hyper-parameters" to denote the algorithm parameters that can be set by the user and are not estimated directly from the data, e.g., the parameter $K$ in the *K-NN* algorithm. In contrast, the term "parameters" in the statistical literature typically refers to the model quantities that are estimated directly by the data, e.g., the weight vector $w$ in a linear regression model $y = w' \cdot x + b$. See Ref. 2 for a definition and discussion too.

[b]The term "overfitting" is a more general term and we prefer the term "overestimating" to characterize this phenomenon.

Table 1. Average estimated accuracy $E(\hat{B})$ when reporting the (estimate) performance of $N$ models with equal true accuracy of 85%. In brackets the 5 and 95 percentilies are shown. The smaller the sample size, and the larger the number of models $N$ out of which selection is performed the larger the overestimation.

| Test Set Sample Size | | | | | | |
|---|---|---|---|---|---|---|
| Number of Models | 20 | 40 | 80 | 100 | 500 | 1000 |
| 5 | 0.935 [0.85; 1.00] | 0.913 [0.85; 0.97] | 0.895 [0.86; 0.94] | 0.891 [0.86; 0.93] | 0.868 [0.85; 0.89] | 0.863 [0.85; 0.88] |
| 10 | 0.959 [0.90; 1.00] | 0.930 [0.90; 0.97] | 0.908 [0.88; 0.94] | 0.902 [0.87; 0.93] | 0.874 [0.86; 0.89] | 0.867 [0.86; 0.88] |
| 20 | 0.977 [0.95; 1.00] | 0.946 [0.90; 0.97] | 0.920 [0.89; 0.95] | 0.913 [0.89; 0.94] | 0.879 [0.87; 0.89] | 0.871 [0.86; 0.88] |
| 50 | 0.993 [0.95; 1.00] | 0.961 [0.93; 1.00] | 0.933 [0.91; 0.96] | 0.925 [0.90; 0.95] | 0.885 [0.87; 0.90] | 0.875 [0.87; 0.88] |
| 100 | 0.999 [1.00; 1.00] | 0.971 [0.95; 1.00] | 0.941 [0.93; 0.96] | 0.932 [0.91; 0.95] | 0.889 [0.88; 0.90] | 0.878 [0.87; 0.89] |
| 1000 | 1.000 [1.00; 1.00] | 0.994 [0.97; 1.00] | 0.962 [0.95; 0.97] | 0.952 [0.94; 0.97] | 0.899 [0.89; 0.91] | 0.885 [0.88; 0.89] |

that exhibits the best *estimated* accuracy $\hat{B}$ *and report this performance as the estimated performance of the selected model*. Let us assume that all models have the same, true accuracy of 85%. What is the expected value of the estimated $\hat{B}$, $E(\hat{B})$ and how biased is it?

Let us denote the accuracy of each model with $P_i = 0, 85$. The *true performance* of the final model is of course also $B = \max P_i = 0, 85$. But, the estimated performance $\hat{B} = \max(\hat{P}_i)$ is biased. Table 1 shows $E(\hat{B})$ for different values of $N$ and $M$ assuming each model makes independent errors on the test set as estimated with 10 000 simulations. The table also shows the 5th and 95th percentile as an indication of the range of the estimation. *Invariably, the expected estimated accuracy* $E(\hat{B})$ *of the final model is overestimated*.

As expected, the bias increases with the number of models tried and decreases with the size of the test set. For sample sizes less than or equal to 100, the bias is significant: when the number of models produced is larger than 100, it is not uncommon to estimate the performance of the best model as 100%. Notice that, when using Cross Validation-based protocols to estimate performance each sample serves once and only once as a test case. Thus, *one can consider the total data-set sample size as the size of the test set*. Typical high-dimensional datasets in biology often contain less than 100 samples and thus, one should be careful with the estimation protocols employed for their analysis.

What about the number of different models tried in an analysis? Is it realistic to expect an analyst to generate thousands of different models? Obviously, it is very rare that any analyst will employ thousands of different algorithms; however, most learning algorithms are parameterized by several different hyper-parameters. For example, the standard 1-norm, polynomial Support Vector Machine algorithm takes as hyper-parameters the

cost $C$ of misclassifications and the degree of the polynomial $d$. Similarly, most variable selection methods take as input a statistical significance threshold or the number of variables to return. If an analyst tries several different methods for imputation, discretization, variable selection, and classification, each with several different hyper-parameter values, the number of combinations explodes and can easily reach into the thousands. Notice that, model selection and optimistic estimation of performance *may also happen unintentionally and implicitly* in many other settings. More specifically, consider a typical publication where a new algorithm is introduced and its performance (after tuning the hyper-parameters) is compared against numerous other alternatives from the literature (again, after tuning their hyper-parameters), on several datasets. The comparison aims to comparatively evaluate the methods. However, *the reported performances of the best method on each dataset suffer from the same problem of multiple inductions and are on average optimistically estimated.*

We now discuss the different factors that affect estimation. In the simulations above, we assume that the $N$ different models provide independent predictions. However, this is unrealistic as the same classifier with slightly different hyper-parameters will produce models that give correlated predictions (e.g., K-NN models with $K = 1$ and $K = 3$ will often make the same errors). Thus, in a real analysis setting, the amount of bias may be smaller than what is expected when assuming no dependence between models. The violation of independence makes the theoretical analysis of the bias difficult and so in this paper, we rely on the empirical evaluations of the different estimation protocols.

There are other factors that affect the bias. For example, the difference of the performance of the best method with the other methods attempted relative to the variance of the estimation, affects the bias. For example, if the best method attempted has a true accuracy of 85% with variance 3% and all the other methods attempted have a true accuracy of 50% with variance 3%, we do not expect considerable bias in the estimation: the best method will always be selected no matter whether its performance is overestimated or underestimated with the specific dataset, and thus on average it will be unbiased. This observation actually forms the basis for the Tibshirani and Tibshirani method[4] described below.

In the remainder of the paper, we revisit the Cross-Validation (CV) protocol. We corroborate[2,5] that CV *overestimates* performance when it is used with hyper-parameter optimization. As expected overestimation of performance increases with decreasing sample sizes. We present three other performance estimation methods in the literature. The first is a simple approach that re-evaluates CV performance by using a different split of the data (CVM-CV).[c] The method by Tibshirani and Tibshirani (hereafter TT)[4] tries to estimate the bias and remove it from the estimation. The Nested Cross Validation (NCV) method[6] cross-validates the whole hyper-parameter optimization procedure which includes an inner cross-validation, hence the name). NCV is a generalization of the technique where data is partitioned in train-validation-test sets.

[c]We thank the anonymous reviewers for suggesting the method.

We show that the behavior of the four methods is markedly different, ranging from the overestimation to conservative estimation of performance bias and variance. To our knowledge, this is the first time these methods are compared against each other on real datasets.

There are two sets of experiments, namely with and without a feature selection pre-processing step. On one side, we expect that the models will gain predictive power from the elimination of irrelevant or superfluous variables. However, the inclusion of one further modelling step increases the number of hyper-parameter configurations to evaluate, and thus performance overestimation should increase as well. Empirically, we show that this is indeed the case. The effect of stratification is also empirically examined. Stratification is a technique that during partitioning of the data into folds forces the same distribution of the outcome classes to each fold. When data are split randomly, on average, the distribution of the outcome in each fold will be the same as in the whole dataset. However, in small sample sizes or imbalanced data it could happen that a fold gets no samples that belong in one of the classes (or in general, the class distribution in a fold is very different from the original). Stratification ensures that this does not occur. We show that stratification has different effects depending on (a) the specific performance estimation method and (b) the performance metric. However, we argue that stratification should always be applied as a cautionary measure against excessive variance in performance.

## 2. Cross-Validation Without Hyper-Parameter Optimization (CV)

K-fold Cross Validation is perhaps the most common method for estimating performance of a learning method for small and medium sample sizes. Despite its popularity, its theoretical properties are arguably not well known especially outside the machine learning community, particularly when it is employed with simultaneous hyper-parameter optimization, as evidenced by the following common machine learning books: Duda (Ref. 7, p. 484) presents CV without discussing it in the context of model selection and only hints that it may underestimate (when used without model selection): "The jackknife [i.e., leave-one-out CV] in particular, generally gives good estimates because each of the $n$ classifiers is quite similar to the classifier being tested …". Similarly, Mitchell[8] (pp. 112, 147, 150) mentions CV but only in the context of hyper-parameter optimization. Bishop[9] does not deal at all with issues of performance estimation and model selection. A notable exception is the Hastie and co-authors[10] book that offers the best treatment of the subject, *upon which the following sections are based.* Yet, CV is still not discussed in the context of model selection.

Let us assume a dataset $D = \{\langle x_i, y_i\rangle, i = 1, \ldots, N\}$, of identically and independently distributed (i.i.d.) predictor vectors $x_i$ and corresponding outcomes $y_i$. Let us also assume that we have *a single method* for learning from such data and producing a single prediction model. *We will denote with* $f(x_i, D)$ *the output of the model produced by the learner f when trained on data D and applied on input* $x_i$. *The actual model produced by f on dataset D is denoted with* $f(\cdot, D)$. We will denote with $L(y, y')$ the loss (error) measure

**Algorithm 1**: K-Fold Cross-Validation $CV(D)$

**Input:** A dataset $D = \{\langle x_i, y_i \rangle, i = 1, \ldots, N\}$
**Output:** A model $M$
An estimation of performance (loss) of $M$

Randomly Partition $D$ to $K$ folds $F_i$
Model $M = f(\cdot, D)$ // the model learned on all data $D$
Estimation $\hat{L}_{CV}$:

$$\hat{e_i} = \frac{1}{N_i} \sum_{j \in F_i} L\left(y_j, f\left(x_j, D_{\setminus i}\right)\right), \hat{L}_{CV} = \frac{1}{K} \sum_{i=1}^{K} \hat{e_i}$$

Return $\langle M, \hat{L}_{CV} \rangle$

of prediction $y'$ when the true output is $y$. One common loss function is the zero-one loss function: $L(y, y') = 1$, if $y \neq y'$ and $L(y, y') = 0$, otherwise. Thus, the average zero-one loss of a classifier equals 1 — *accuracy*, i.e., it is the probability of making an incorrect classification. K-fold CV partitions the data $D$ into $K$ subsets called folds $F_i, \ldots, F_k$. We denote with $D_{\setminus i}$ the data excluding fold $F_i$ and $N_i$ the sample size of each fold. The K-fold CV algorithm is shown in Algorithm 1.

First, notice that *CV should return the model learned from all data D,* $f(\cdot, D)$.[d] *This is the model to be employed operationally for classification.* It then tries to estimate the performance of the returned model by constructing $K$ other models from datasets $D_{\setminus i}$, each time excluding one fold from the training set. Each of these models is then applied on each fold $F_i$ serving as test and the loss is averaged over all samples.

Is $\hat{L}_{CV}$ an unbiased estimate of the loss of $f(\cdot, D)$? First, notice that each sample $x_i$ is used once and only once as a test case. *Thus, effectively there are as many i.i.d. test cases as samples in the dataset*. Perhaps, this characteristic is what makes the CV so popular versus other protocols such as repeatedly partitioning the dataset to train-test subsets. The test size being as large as possible could facilitate the estimation of the loss and its variance (although, theoretical results show that there is no unbiased estimator of the variance for the CV![11]). However, test cases are predicted with different models! If these models were trained on independent train sets of size equal to the original data $D$, then CV would indeed estimate the average loss of the models produced by the specific learning method on the specific task when trained with the specific sample size. As it stands though, since the models are correlated and have smaller size than the original we can state the following:

> *K-Fold CV estimates the average loss of models returned by the specific learning method f on the specific classification task when trained with subsets of D of size* $|D_{\setminus i}|$.

[d]This is often a source of confusion for some practitioners who sometimes wonder which model to return out of the ones produced during Cross-Validation.

**Algorithm 2:** K-Fold Cross-Validation with Hyper-Parameter Optimization (Model Selection) $CVM(D, \boldsymbol{a})$

**Input:** A dataset $D = \{\langle x_i, y_i\rangle, i = 1, \ldots, N\}$
A set of hyper-parameter value combinations $\boldsymbol{a}$
**Output:** A model $M$
An estimation of performance (loss) of $M$

Partition $D$ to $K$ folds $F_i$
Estimate $\hat{L}_{CV}(a)$ for each $a \in \boldsymbol{a}$:

$$\hat{e}_i(a) = \frac{1}{N_i}\sum_{j \in F_i} L\big(y_j, f(x_j, D_{\setminus i}, a)\big),\ \hat{L}_{CV}(a) = \frac{1}{K}\sum_{i=1}^{K} \hat{e}_i(a)$$

Find minimizer $a^*$ of $\hat{L}_{CV}(a)$ // "best hyper-parameters"
$M = f(\cdot, D, a^*)$ // the model from all data $D$ with the best hyper-parameters
$\hat{L}_{CVM} = \hat{L}_{CV}(a^*)$
Return $\langle M, \hat{L}_{CVM}\rangle$

Since $|D_{\setminus i}| = (K-1)/K \cdot |D| < |D|$ (e.g., for 5-fold, we are using 80% of the total sample size for training each time) and *assuming that the learning method improves on average with larger sample sizes* we expect $\hat{L}_{CV}$ to be conservative (i.e., the true performance be underestimated). Exactly how conservative it will be depends on where the classifier is operating on its learning curve for this specific task. It also depends on the number of folds $K$: the larger the $K$, the more ($K$-1)/$K$ approaches 100% and the bias disappears, i.e., leave-one-out CV should be the least biased (however, there may be still be significant estimation problems, see Ref. 12, p. 151, and Ref. 5 for an extreme failure of leave-one-out CV). When sample sizes are small or distributions are imbalanced (i.e., some classes are quite rare in the data), we expect most classifiers to quickly benefit from increased sample size, and thus $\hat{L}_{CV}$ to be more conservative.

## 3. Cross-Validation With Hyper-Parameter Optimization (CVM)

A typical data analysis involves several steps (representing the data, imputation, discretization, variable selection or dimensionality reduction, learning a classifier) each with hundreds of available choices of algorithms in the literature. In addition, each algorithm takes several hyper-parameter values that should be tuned by the user. A general method for tuning the hyper-parameters is to try a set of *predefined* combinations of methods and corresponding values and select the best. We will represent this set with a set $\boldsymbol{a}$ containing hyper-parameter values, e.g, $\boldsymbol{a}$ = {⟨no variable selection, K-NN, K = 5⟩, ⟨Lasso, λ = 2, linear SVM, C = 10⟩} when the intent is to try K-NN with no variable selection, and a linear SVM using the Lasso algorithm for variable selection. The pseudo-code is shown in Algorithm 2. The symbol $f(x, D, a)$ now denotes the output of the model learned when using hyper-parameters $a$ on dataset $D$ and applied on input $x$. Correspondingly, the symbol $f(\cdot, D, a)$ denotes the model produced by applying hyper-parameters $a$ on $D$.

The quantity $L_{CV}(a)$ is now parameterized by the specific values $a$ and the minimizer of the loss (maximizer of performance) $a^*$ is found. The final model returned is $f(\cdot, D, a^*)$, i.e., the model produced by setting hyper-parameter values to $a^*$ and learning from *all data D*.

On one hand, we expect CV with model selection (hereafter, *CVM*) to underestimate performance because estimations are computed using models trained on only a subset of the dataset. On the other hand, we expect *CVM* to overestimate performance because it returns the maximum performance found after trying several hyper-parameter values. In Section 8 we examine this behavior empirically and determine (in concordance with Refs. 2 and 5) that indeed when sample size is relatively small and the number of models tried is in the hundreds CVM overestimates performance. Thus, in these cases other types of estimation protocols are required.

## 4. The Double Cross Validation Method (CVM-CV)

The CVM is biased because when trying hundreds or more learning methods, what appears to be the best one has probably also been "lucky" for the particular test sets. Thus, one idea to reduce the bias is to re-evaluate the selected, best method on different test sets. Of course, since we are limited to the given samples (dataset) it is impossible to do so on truly different test cases. One idea thus, is to re-evaluate the selected method on a different split (partitioning) to folds and repeat Cross-Validation only for the single, selected, best method. We name this approach CVM-CV, since it sequentially performs CVM and CV for model selection and performance estimation, respectively and it is shown in Algorithm 3. The tilde symbol "~" is used to denote a returned value that is ignored. Notice that, CVM-CV is not theoretically expected to fully remove the over-estimation bias: information from the test sets in the final Cross-Validation step for performance estimation is still employed during training to select the best model. Nevertheless, our experiments

**Algorithm 3**: Double Cross Validation $CVM - CV(D, \boldsymbol{a})$

**Input:** A dataset $D = \{\langle x_i, y_i \rangle, i = 1, \dots, N\}$
A set of hyper-parameter value combinations $\boldsymbol{a}$
**Output:** A model $M$
An estimation of performance (loss) of $M$

Partition $D$ to $K$ folds $F_i$
$\langle M, \sim \rangle = CVM(D, \boldsymbol{a})$
$a^*$ is the parameter configuration corresponding to $M$
Estimation $\hat{L}_{CVM-CV}$:
Partition $D$ to $K$ new randomly chosen folds $F_j$
$\langle \sim, \hat{L}_{CVM-CV} \rangle = CVM(D, a^*)$
Return $\langle M, \hat{L}_{CVM-CV} \rangle$

**Algorithm 4:** $TT(D, \boldsymbol{a})$

**Input:** A dataset $D = \{\langle x_i, y_i\rangle, i = 1, \ldots, N\}$
A set of hyper-parameter value combinations $\boldsymbol{a}$
**Output:** A model $M$
An estimation of performance (loss) of $M$

Partition $D$ to $K$ folds $F_i$
$\langle M, \sim\rangle = CVM(D, \boldsymbol{a})$

$$\hat{e}_i(a) = \frac{1}{N_i}\sum_{j\in F_i} L\big(y_j, f(x_j, D_{\setminus i}, a)\big),\ \hat{L}_{CV}(a) = \frac{1}{K}\sum_{i=1}^{K}\widehat{e_i(a)}$$

Find minimizer $a^*$ of $\hat{L}_{CV}(a)$ // global optimizer
Find minimizers $a_k$ of $e_k(a)$ // the minimizers for each fold

Estimate $\widehat{Bias} = \frac{1}{K}\sum_{k=1}^{K}(\, e_k(a^*) - \, e_k(a_k))$

$M = f(\cdot, D, a^*)$, i.e., the model learned on all data $D$ with the best hyper-parameters
$\hat{L}_{TT} = \hat{L}_{CV}(a^*) + \widehat{Bias}$
Return $\langle M, \hat{L}_{TT}\rangle$

show that this relatively computationally-efficient approach does reduce CVM over-estimation bias.

## 5. The Tibshirani and Tibshirani (TT) Method

The TT method[4] attempts to *heuristically and approximately* estimate the bias of the CV error estimation due to model selection and add it to the final estimate of loss. For each fold, the bias due to model selection is estimated as $e_k(a^*) - e_k(a_k)$ where, as before, $e_k$ is the average loss in fold $k$, $a_k$ is the hyper-parameter values that minimizes the loss for fold $k$, and $a^*$ the global minimizer over all folds. Notice that, if in all folds the same values $a_k$ provide the best performance, then these will also be selected globally and hence $a_k = a^*$ for $k = 1, \ldots, K$. In this case, *the bias estimate will be zero*. The justification of this estimate for the bias is in Ref. 4. It is quite important to notice that TT does not require any additional model training and has minimum computational overhead.

## 6. The Nested Cross-Validation Method (NCV)

We could not trace who introduced or coined up first the name Nested Cross-Validation (NCV) method but the authors have independently discovered it and using it since 2005;[6,13,14] one early comment hinting of the method is in Ref. 15, while Witten and Frank briefly discuss the need of treating any parameter tuning step as part of the training process when assessing performance (see Ref. 12, p. 286).

A similar method in a bioinformatics analysis was used as early as 2003.[16] *The main idea is to consider the model selection as part of the learning procedure f.* Thus, *f* tests

several hyper-parameter values, selects the best using CV, and returns a *single model*. NCV *cross-validates f* to estimate the performance of the average model returned by *f* just as normal CV would do with any other learning method taking no hyper-parameters; it's just that *f* now contains an internal CV trying to select the best model. NCV is a generalization of the Train-Validation-Test protocol where one trains on the Train set for all hyper-parameter values, selects the ones that provide the best performance on Validation, trains on Train+Validation a *single model* using the best-found values and estimates its performance on Test. Since Test is used only once by a single model, performance estimation has no bias due to the model selection process. The final model is trained on *all data* using the best found values for *a*. NCV generalizes the above protocol to cross-validate every step of this procedure: for each Test, all folds serve as Validation, and this process is repeated for each fold serving as Test. The pseudo-code is shown in Algorithm 5. The pseudo-code is similar to CV (Algorithm 1) with CVM (Cross-Validation with Model Selection, Algorithm 2) serving as the learning function *f*. NCV requires a quadratic number of models to be trained to the number of folds *K* (one model is trained for every possible pair of two folds serving as test and validation respectively) and thus it is the most computationally expensive protocol out of the four.

**Algorithm 5:** K-Fold Nested Cross-Validation $NCV(D, \boldsymbol{a})$

**Input:** A dataset $D = \{\langle x_i, y_i \rangle, i = 1, \ldots, N\}$
A set of hyper-parameter value combinations $\boldsymbol{a}$
**Output:** A model $M$
An estimation of performance (loss) of $M$

Partition $D$ to $K$ folds $F_i$
$\langle M, \sim \rangle = CVM(D, \boldsymbol{a})$
Estimation $\hat{L}_{NCV}$:
$\langle M_i, \cdot \rangle = CVM\left(D_{\setminus i}, \boldsymbol{a}\right)$\\best performing model on $D_{\setminus i}$
$\hat{e_i} = \frac{1}{N_i} \sum_{j \in F_i} L\left(y_j, M_i\left(x_j, D_{\setminus i}\right)\right)$, $\hat{L}_{NCV} = \frac{1}{K} \sum_{i=1}^{K} \hat{e_i}$
Return $\langle M, \hat{L}_{NCV} \rangle$

## 7. Stratification of Folds

In CV, folds are partitioned randomly which should maintain *on average* the same class distribution in each fold. However, in cases of small sample sizes or highly imbalanced class distributions it may happen that some folds contain no samples from one of the classes (or in general, the class distribution is very different from the original). In that case, the estimation of performance for that fold will exclude that class. To avoid this case, "in stratified cross-validation, the folds are stratified so that they contain approximately the same proportions of labels as the original dataset".[5] Notice that leave-one-out

CV guarantees that each fold will be unstratified since it contains only one sample which can cause serious estimation problems[5] (Ref. 12, p. 151).

## 8. Empirical Comparison of Different Protocols

We performed an empirical comparison in order to assess the characteristics of each data-analysis protocol. Particularly, we focus on three specific aspects of the protocols' performances: (a) bias and variance of the estimation, (b) effect of feature selection and (c) effect of stratification.

*Notice that, assuming data are partitioned into the same folds, all methods return the same model*, that is, the model returned by $f$ on *all data* $D$ using the minimizer $a^*$ of the CV error. However, each methods return a different estimate of the performance for this model.

### 8.1. *The experimental set-up*

**Original Datasets:** Five datasets from different scientific fields were employed for the experimentations. The computational task for each dataset consists in predicting a binary outcome on the basis of a set of numerical predictors (binary classification). Datasets were selected to have a relatively large number of samples so that several smaller datasets that follow the same joint distribution and can be sub-sampled from the original dataset; when the number of sample size is large the sub-samples are relatively independent providing independent estimates of all metrics in the experimental part. In more detail the **SPECT**[17] dataset contains data from Single Photon Emission Computed Tomography images collected in both healthy and cardiac patients. Data in **Gamma**[18] consist of simulated registrations of high energy gamma particles in an atmospheric Cherenkov telescope, where each gamma particle can be originated from the upper atmosphere (background noise) or being a primary gamma particle (signal). Discriminating biodegradable versus non-biodegradable molecules on the basis of their chemical characteristics is the aim of the **Biodeg**[19] dataset. The **Bank**[20] dataset was gathered by direct marketing campaigns (phone calls) of a Portuguese banking institution for discriminating customers who want to subscribe a term deposit and those who do not. Last, **CD4vsCD8**[21] contains the phosphorylation levels of 18 intra-cellular proteins as predictors to discriminate naïve CD4+ and CD8+ human immune system cells. **SeismicBump**[22] focuses on forecasting high energy (higher than $10^4$ J) in coal mines. Data come from longwalls located in a Polish coal mine. The **MiniBoone** dataset is taken from the first phase of the Booster Neutrino Experiment conducted in the FermiLab;[23] the goal is to distinguish between electron neutrinos (signal) and muon neutrinos (background). Table 2 summarizes datasets' characteristics. It should be noticed that the outcome distribution considerably varies across datasets.

**Model Selection:** To generate the hyper-parameter vectors in $\boldsymbol{a}$ we employed two different strategies, named *No Feature Selection* (*NFS*) and *With Feature Selection* (*WFS*).

Table 2. Datasets' characteristics. $D_{pool}$ is a 30% partition from which sub-sampled datasets are produced. $D_{hold\text{-}out}$ is the remaining 70% of samples from which an accurate estimation of the true performance is computed.

| Dataset Name | # Samples | # Attributes | Classes Ratio | $\|D_{pool}\|$ | $\|D_{hold\text{-}out}\|$ | Reference |
|---|---|---|---|---|---|---|
| SPECT | 267 | 22 | 3.85 | 81 | 186 | 17 |
| Biodeg | 1055 | 41 | 1.96 | 317 | 738 | 19 |
| SeismicBumps | 2584 | 18 | 14.2 | 776 | 1808 | 22 |
| Gamma | 19020 | 11 | 1.84 | 5706 | 13314 | 18 |
| CD4vsCD8 | 24126 | 18 | 1.13 | 7238 | 16888 | 21 |
| MiniBooNE | 31104 | 50 | 2.55 | 9332 | 21772 | 23 |
| Bank | 45211 | 17 | 7.54 | 13564 | 31647 | 20 |

*Strategy NFS* (*No Feature Selection*) includes three different modelers: the Logistic Regression classifier (Ref. 9, p. 205), as implemented in Matlab 2013b, that takes no hyper-parameters; the Decision Tree,[24] as implemented also in Matlab 2013b with hyper-parameters MinLeaf and MinParents both within {1, 2,…,10, 20, 30, 40, 50}; Support Vector Machines as implemented in the libsvm software[25] with linear, Gaussian ($\gamma \in \{0.1,\ 1,\ 10\}$) and polynomial (degree $d \in \{2,\ 3,\ 4\}$, $\gamma \in \{0.1,\ 1,\ 10\}$) kernels, and cost parameter $C \in \{0.1,\ 1,\ 10\}$. When a classifier takes multiple hyper-parameters, all combinations of choices are tried. Overall, 247 hyper-parameter value combinations and corresponding models are produced each time to select the best.

*Strategy WFS* (*With Feature Selection*) adds feature selection methods as preprocessing steps to Strategy *NFS*. Two feature selection methods are tried each time, namely the univariate selection and the Statistically Equivalent Signature (SES) algorithm.[26] The former simply applies a statistical test for assessing the association between each individual predictor and the target outcome (chi-square test for categorical variables and Student *t*-test for continuous ones). Predictors whose *p*-values are below a given significance threshold *t* are retained for successive computations. The SES algorithm[26] belongs to the family of constraint-based, Markov-Blanket inspired feature selection methods.[27] In short, SES repetitively applies a statistical test of conditional independence for identifying the set of predictors that are associated with the outcome *given any combination of the remaining variables*. Also SES requires the user to set a priori a significance threshold *t*, along with the hyper-parameter *maxK* that limits the number of predictors to condition upon. Both feature selection methods are coupled in turn with each modeler in order to build the hyper-parameters vector $\boldsymbol{a}$. The significance threshold *t* is varied in {0.01, 0.05} for both methods, while *maxK* varies in {3, 5}, bringing the number of hyper-parameter combinations produced in Strategy *WFS* to 1729.

**Sub-Datasets and Hold-out Datasets.** Each original dataset *D* is partitioned into two separate, stratified parts: $D_{pool}$, containing 30% of the total samples, and the hold-out set $D_{hold\text{-}out}$, consisting of the remaining samples. Subsequently, for each $D_{pool}$ *N* sub-datasets

are randomly sampled with replacement for each sample size in the set {20, 40, 60, 80, 100, 500 and 1500}, for a total of $5 \times 7 \times N$ sub-datasets $D_{i,j,k}$ (where $i$ indexes the original dataset, $j$ the sample size, and $k$ the sub-sampling). Most of the original datasets have been selected with a relatively large sample size so that *each* $D_{hold\text{-}out}$ *is large enough to allow an accurate* (*low variance*) *estimation of performance*. In addition, the size of $D_{pool}$ is also relatively large so that each sub-sampled dataset *to be approximately considered a dataset independently sampled from the data population of the problem.* Nevertheless, we also include a couple of datasets with smaller sample size. We set the number of sub-samples to be $N = 30$.

**Bias and Variance of each Protocol:** For each of the data analysis protocols *CVM, CVM-CV, TT,* and *NCV* both the stratified and the non-stratified versions are applied to each sub-dataset, in order to select the "best model/hyper-parameter values" and estimate its performance $\hat{L}$. For each sub-dataset, the same split in $K = 10$ folds was employed for the stratified versions of *CVM, CVM-CV, TT and NCV*, so that the three data-analysis protocols always select exactly the same model, and differ only in the estimation of performance. For the NCV, the internal CV loop uses $K' = K - 1$ folds. Some of the dataset though are characterized by a particular high-class ratio, and typically this leads to a scarcity of instances of the rarest class in some sub-datasets. If the number $R$ of instances of a given class is smaller than $K$, we set $K = R$ in order to ensure the presence of both classes in each fold. For *NCV* and *NS-NCV*, we forgo to analyze sub-datasets where $R < 3$.

The bias is computed as $L_{hold-out} - \hat{L}$. Thus, *a positive bias indicates a higher* "*true*" *error* (i.e., as estimated on the hold-out set) than the one estimated by the corresponding analysis protocol *and implies the estimation protocol is optimistic*. For each protocol, original dataset, and sample size the true mean bias, its variance and its standard deviation are computed over 30 sub-samplings.

**Performance Metric:** All algorithms are presented using a loss function $L$ computed for each sample and averaged out for each fold and then over all folds. The *zero-one loss* function is typically assumed corresponding to 1-accuracy of the classifier. A valid alternative as metric for binary classification problems the Area Under the Receiver's Operating Characteristic Curve ($AUC$).[28] The AUC does not depend on the prior class distribution. In contrast, the zero-one loss depends on the class distribution: for a problem with class distribution of 50–50%, a classifier with accuracy 85% (loss 15%) has greatly improved over the baseline of a trivial classifier predicting the majority class; for a problem of 84 – 16% class distribution, a classifier with 85% accuracy has not improved much over the baseline. On the other hand, computing AUC on small test sets leads to poor estimates,[29] and it is impossible when leave-one-out cross validation is used (unless multiple predictions are pooled together, a practice that creates additional issues[29]). Moreover, the AUC cannot be expressed as a loss function $L(y, y')$ where $y'$ is a single prediction. Nevertheless, all Algorithms 1–4 remain the same if we substitute $\hat{e}_i = 1 - AUC\left(f\left(\cdot,\ D_{\setminus i}\right), F_i\right)$, i.e., the error in fold $i$ is 1 minus the AUC of the model learned by $f$ on all data except fold $F_i$, as estimated on $F_i$ as the test set. In order to contrast the

properties of the two metrics, we have performed all analyses twice, using in turn 0–1 loss and 1-AUC as the metrics to optimize. For both metrics a positive bias corresponds to overestimated performance.

### 8.2. *Experimental results*

The results of the analysis greatly differ depending on the specific dataset, performance metric and performance estimation method. The following Tables and Figures show the results obtained with the AUC metric, while the remaining results are provided in Appendix A.

We first comment the results obtained in the experimentation following Strategy *NFS*. Fig. 1 (first row) shows the average loss bias of the four methods, (stratified version) suggesting that indeed *CVM overestimates performance* for small sample sizes (underestimates error) corroborating the results in Refs. 2 and 5. In contrast, *NCV tends to be over pessimistic* and to underestimate the real performances. *CVM-CV and TT exhibit results*

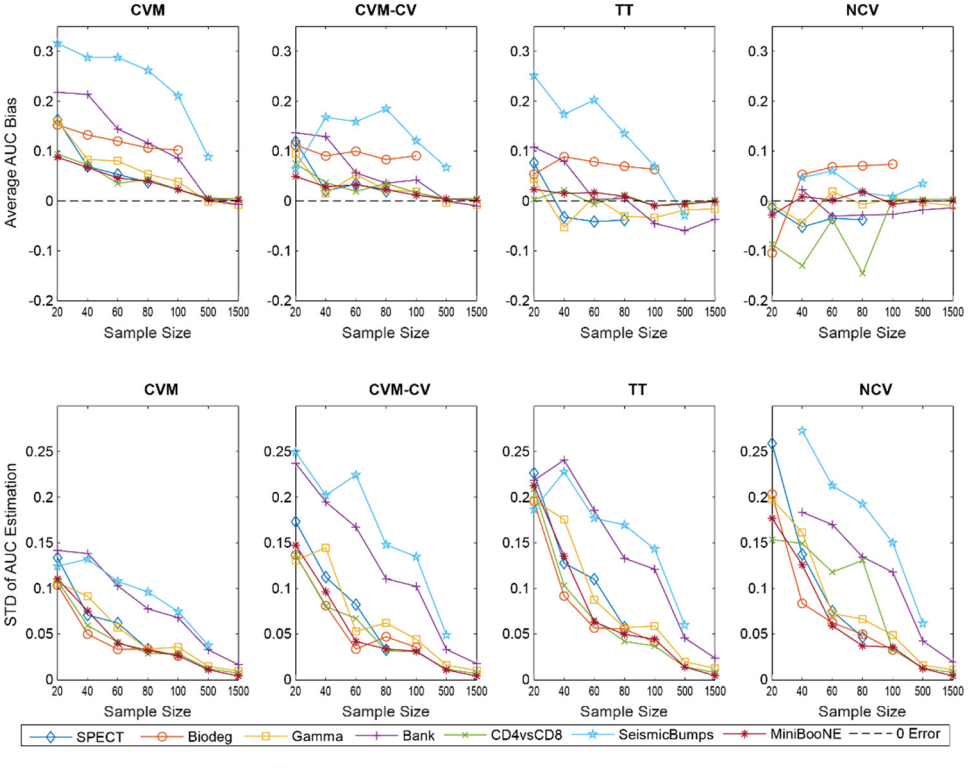


Fig. 1. (Color online) Average loss and variance for AUC metric in Strategy NFS. From left to right: stratified CVM, CVM-CV, TT, and NCV. Top row contains average bias, second row the standard deviation of performance estimation. The results largely vary depending on the specific dataset. In general, CVM is clearly optimistic (positive bias) for sample sizes less or equal to 100, while NCV tends to underestimate performances. CVM-CV and TT show a behavior that is in between these two extremes. CVM has the lowest variance, at least for small sample sizes.

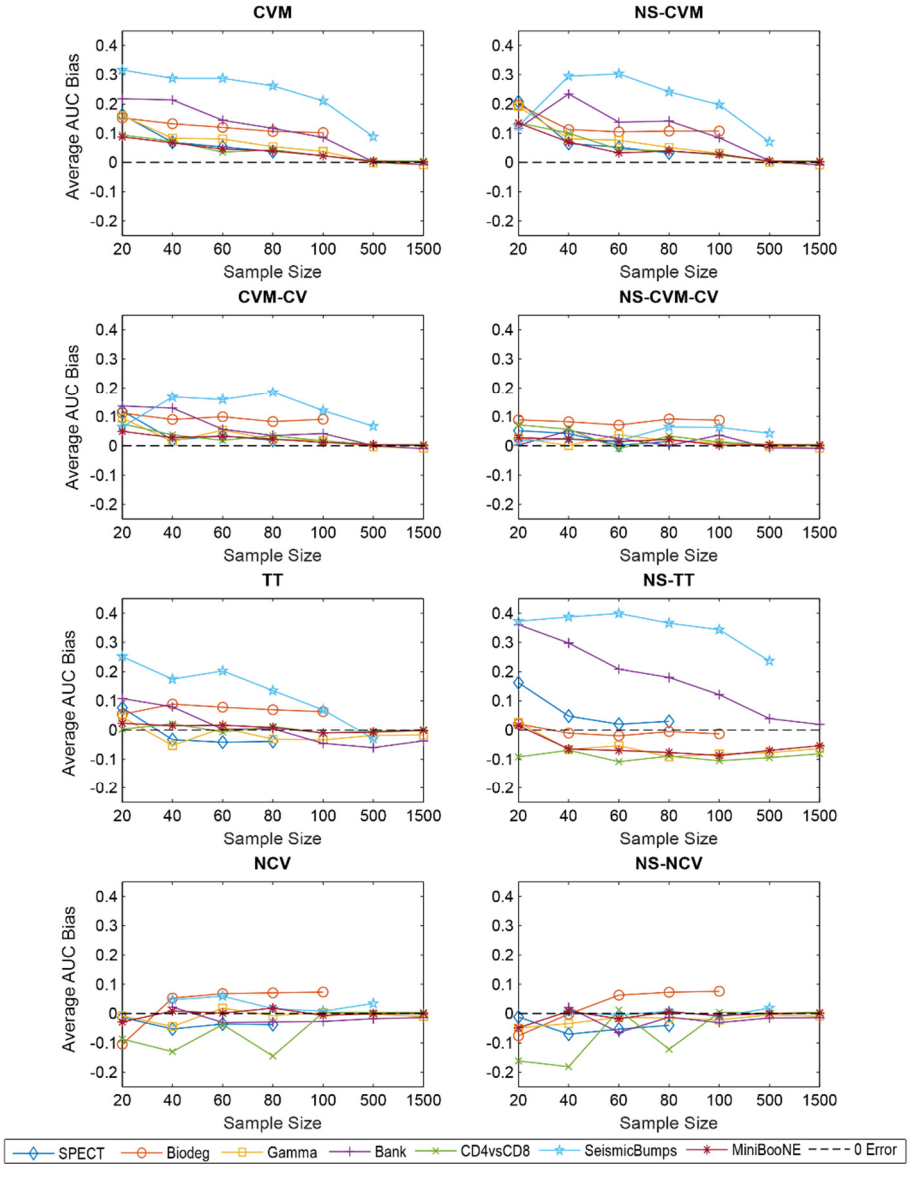


Fig. 2. (Color online) Effect of stratification using the AUC in Strategy NFS. The first column reports the average bias for the stratified versions of each method, while the second column for the non-stratified ones. The effect of stratification is dependent on each specific method and dataset.

*that are between these two extremes, with CVM > CVM-CV > TT > NCV in terms of overestimation.* It should be noted that the results on the SeismicBump dataset strongly penalize the CVM, CVM-CV and TT methods. Interestingly, this dataset has the highest ratio between outcome classes (14.2), suggesting that estimating AUC performances in highly imbalanced datasets may be challenging for these three methods.

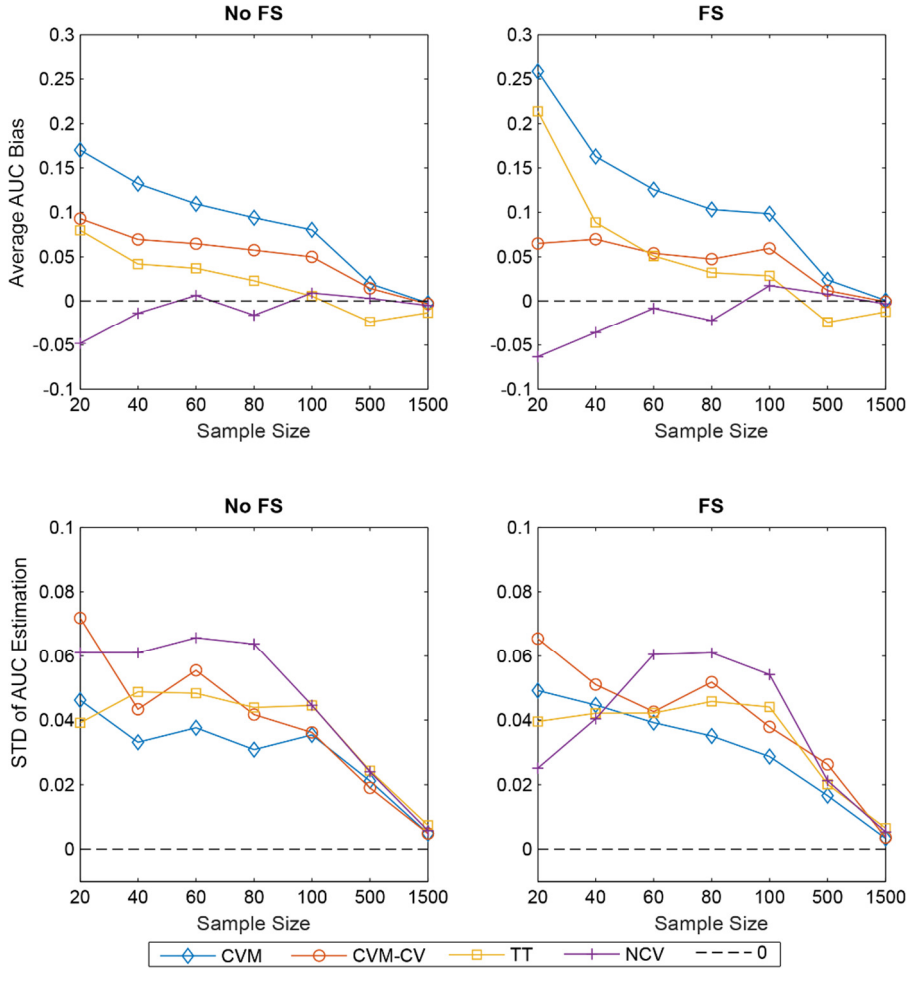


Fig. 3. (Color online) Comparing Strategy NFS (No Feature Selection) and WFS (With Feature Selection) over AUC. The first row reports the average bias of each method for Strategy NFS and WFS, respectively, while the second row provides the variance in performance estimation. CVM and TT shows an evident increment in bias in Strategy WFS, presumably due to the larger hyper-parameter space explored in this setting.

Table 3 shows the bias averaged over all datasets, where it is shown that CVM overestimates AUC up to ~17 points for small sample sizes, while CVM-CV and TT are always below 10 points, and NCV bias is never outside the range of 5 points. We perform a *t*-test for the null hypothesis that the bias is zero, which is typically rejected: all methods usually exhibit some bias whether positive or negative.

The second row of Fig. 1 and Table 4 show the standard deviations for the loss bias. We apply the O'Brien's modification of Levene's statistical test[30] with the null hypothesis that the variance of a method is the same as the corresponding variance for the same sample size as the NCV. We note that CVM has the lowest variance, while all the other methods show almost statistically indistinguishable variances.

Table 3. Average AUC Bias over Datasets (Strategy *NFS*). *P*-values produced by a *t*-test with null hypothesis the mean bias is zero ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.1702** | 0.1581** | 0.0929** | 0,0407** | 0.0798** | 0.1237** | –0.0483* | –0.0696** |
| 40 | 0.1321** | 0.1367** | 0.0695** | 0,0398** | 0.0418** | 0.0744** | –0.0137 | –0.0364* |
| 60 | 0.1095** | 0.1072** | 0.0647** | 0,0223** | 0.0371** | 0.0538* | 0.0065 | –0.0113 |
| 80 | 0.0939** | 0.0933** | 0.0574** | 0,0348** | 0.023** | 0.0447 | –0.0162 | –0.0147 |
| 100 | 0.0803** | 0.0788** | 0.0499** | 0,0351** | 0.0056 | 0.0296 | 0.0093* | 0.0017 |
| 500 | 0.0197** | 0.0172** | 0.0143** | 0,0079* | –0.0236** | 0.0068 | 0.0031 | 0.0002 |
| 1500 | –0.0023 | –0.0024 | –0.0031 | –0,0028 | –0.0132** | –0.0447** | –0.0049** | –0.0044** |

Table 4. Standard deviation of AUC estimations over Datasets (Strategy *NFS*). *P*-values produced by a test with null hypothesis that the variances are the same as the corresponding variance of the NCV protocol ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.1289** | 0.1681** | 0.2043 | 0,2113 | 0.214 | 0.1485** | 0.2073 | 0.2156 |
| 40 | 0.1063** | 0.0991** | 0.1571* | 0,1871 | 0.1872 | 0.1026** | 0.1946 | 0.1908 |
| 60 | 0.0845** | 0.0881** | 0.1435 | 0,1729 | 0.1439 | 0.0787** | 0.1637 | 0.1855 |
| 80 | 0.0711** | 0.0769** | 0.1057** | 0,1493 | 0.124* | 0.0742** | 0.1544 | 0.156 |
| 100 | 0.0757** | 0.0806** | 0.1136* | 0,1352 | 0.1342 | 0.0659** | 0.1474 | 0.1498 |
| 500 | 0.0758** | 0.0745** | 0.0834 | 0,0867 | 0.1182** | 0.0436** | 0.0967 | 0.0938 |
| 1500 | 0.0436 | 0.0441 | 0.0448 | 0,0444 | 0.0542* | 0.0298** | 0.0458 | 0.046 |

Table 5. Average AUC on the hold-out sets (Strategy *NFS*). All methods use CVM for model selection and thus have the same performances on the hold-out sets. NS stands for Non-Stratified.

| | CVM | NS-CVM |
|---|---|---|
| 20 | 0.6903 | 0.6720 |
| 40 | 0.7385 | 0.7377 |
| 60 | 0.7753 | 0.7783 |
| 80 | 0.7934 | 0.7898 |
| 100 | 0.8015 | 0.8004 |
| 500 | 0.8555 | 0.8604 |
| 1500 | 0.9163 | 0.9163 |

Table 5 reports the average performances on the hold-out set. As expected, these performances improve with sample size, because the final models are trained on a larger number of samples. The corresponding results for the accuracy metric are reported in Fig. 4 and Tables 6–8 in Appendix A, and generally follow the same patterns and

conclusions as the ones reported above for the AUC metric. The only noticeable difference is a large improvement in the average bias for the SeismicBump dataset. A close inspection of the results for this dataset reveals that all methods tend to select models with performances close to the trivial classifier, both during the model selection and the performance estimation on the hold-out set. The selection of these models minimizes the bias but they have little practical utility, since they tend to predict the most frequent class. This example clearly underlines accuracy's inadequacy for tasks involving highly imbalanced datasets.

Figure 2 contrasts methods' stratified and non-stratified versions for Strategy *NFS* and AUC. The effect of stratification seems to be quite dependent on the specific method. The non-stratified version of NCV has larger bias and variance than the stratified version for small sample sizes, while for other protocols the non-stratified version shows a decreased bias at the cost of larger variance (see Tables 3 and 4). Interestingly, the results for the accuracy metric show an almost identical pattern (see Fig. 5, Tables 6 and 7 in Appendix A). In general, we do not suggest the use of non-stratification, given the increase of variance that usually provides.

Finally, Fig. 3 shows the effect of feature selection in the analysis and contrasts Strategy NFS and Strategy WFS on the AUC metric. The average bias for both CVM and TT increases in Strategy WFS. This increment is explained by the fact that Strategy WFS explores a larger hyper-parameter space than Strategy NFS. The lack of increment in predictive power in Strategy WFS is probably due to absence of irrelevant variables: all datasets have a limited dimensionality (max number of features: 40). In terms of variance the NCV method shows a decrease in standard deviation for small sample sizes in the experimentation with feature selection. Similar results are observed with the accuracy metric (Fig. 6), where the decrease in variance is present for all the methods.

## 9. Related Work and Discussion

Estimating performance of the final reported model while simultaneously selecting the best pipeline of algorithms and tuning their hyper-parameters is a fundamental task for any data analyst. Yet, arguably these issues have not been examined in full depth in the literature. The origins of cross-validation in statistics can be traced back to the "jackknife" technique of Quenouille[31] in the statistical community. In machine learning,[5] studied the cross-validation without model selection (the title of the paper may be confusing) comparing it against the bootstrap and reaching the important conclusion that (a) CV is preferable to the bootstrap, (b) a value of $K = 10$ is preferable for the number of folds versus a leave-one-out, and (c) stratification is also always preferable. In terms of theory, Bengio[11] showed that there exist no unbiased estimator for the variance of the CV performance estimation, which impact hypothesis testing of performance using the CV.

To the extent of our knowledge the first to study the problem of bias *in the context of model selection* in machine learning is Ref. 3. Varma[32] demonstrated the optimism of the CVM protocol and instead suggests the use of the NCV protocol. Unfortunately, all their experiments are performed on simulated data only. Tibshirani and Tibshirani[4] introduced

the TT protocol but they do not compare it against alternatives and they include only a proof-of-concept experiment on a single dataset. Thus, the present paper is the first work that compares all four protocols (CVM, CVM-CV, NCV, and TT) on multiple real datasets.

Based on our experiments we found evidence that both the CVM-CV and the TT method have relatively small bias for sample sizes above 20 and have about the same variance as the NCV; the TT method does not introduce additional computational overhead. However, the TT method seems to overestimate in very small sample sizes when a large number of hyper-parameter configurations are tested. Moreover, caution should still be exercised and further research is required to better investigate the properties of the TT protocol. One particularly worrisome situation is the use of TT in a leave-one-out fashion which we advise against. In this extreme case, each fold contains a single test case. If the overall-best classifier predicts it wrong, the loss is 1. If any other classifier tried predicts it correctly its loss is 0. When numerous classifiers are tried at least one of them will predict the test case correctly with high probability. In this case, the estimation of the bias $\widehat{Bias} = 1/K \sum_{k=1}^{K}( e_k(a^*) - e_k(a_k))$ of the TT method will be equal to the loss of the best classifier. Thus, the TT will estimate the loss of the best classifier found as $\hat{L}_{CV}(a^*) + \widehat{Bias} = 2 \cdot \hat{L}_{CV}(a^*)$, i.e., twice as much as found during leave-one-out CV. To recapture: in leave-one-out CV when the number of classifiers tried is high, TT estimates the loss of the best classifier found as twice its cross-validated loss, which is overly conservative.

A previous version of this work has appeared elsewhere,[33] presenting a more restricted methodological discussion and empirical comparison. Moreover, the analysis protocol of the present version markedly differs from the protocol of the previous one. In Ref. 33 the class ratio for sub-datasets with sample size $\leq 80$ was forced to one, i.e., the sub-sampling procedure was selecting an equal number of instances from the two classes, independently by the original class distribution. In the present experimentations the original class distribution is maintained in the sub-datasets, and the number of folds is dynamically changed in order to ensure at least one instance from the rarest class in each fold. This important change originates a number of differences in the results of the two works. We do underline though that the findings and conclusions of the previous study are still valid in the context of the design of its experimentations.

We also note *the concerning issue* that the variance of estimation for small sample sizes is large, again in concordance with the experiments in Ref. 2. The authors in the latter advocate methods that may be biased but exhibit reduced variance. However, we believe that CVM is too biased no matter its variance; implicitly the authors in Ref. 2 agree when they declare that model selection should be integrated in the performance estimation procedure in such a way that test samples are never employed for selecting the best model. Instead, they suggest as alternatives limiting the extent of the search of the hyper-parameters or performing model averaging. In our opinion, neither option is satisfactory for all analysis purposes and more research is required. One approach that we suggest is to repeat the whole analysis several times using a different random partitioning

to folds each time, and average the loss estimations. Repeating the analysis for different fold partitionings can be performed both for the inner CV loop (if one employs NCV) or just the outer CV loop. Averaging over several repeats reduces the component of the variance that is due to the specific partition to folds, which could be relatively high for small sample sizes.

Another source of variance of performance estimation is due to the stochastic nature of certain classification algorithms. Numerous classification methods are non-deterministic and will return a different model even when trained on the same dataset. Typical examples include Artificial Neural Networks (ANNs), where the final model typically depends on the random initialization of weights. The effect of initialization may be quite intense and result into very different models returned. The exact same theory and algorithms presented above apply to such classifiers; however, one would expect an even larger variance of estimated performances because an additional variance component is added due to the stochastic nature of the classifier training. In this case, we would suggest training the same classifier multiple times and averaging the results to produce an estimate of performance.

Particularly, for ANNs we note further possible complications when using the above protocols. Let us assume that the number of epochs of weight updates is used as a hyper-parameter in the above protocols. In this case, the value of the number of epochs that optimizes the CV loss is selected, and then used to train an ANN on the full dataset. However, training the ANN on a larger sample size may require many more epochs to achieve a good fit on the dataset. Using the same number of epochs as in a smaller dataset may underfit and result in high loss. This violates the assumption made in Section 2 that "*the learning method improves on average with larger sample sizes*". The number-of-epochs hyper-parameter is highly depended on the sample size and thus possibly violates this assumption. To satisfy the assumption it should be the case that training the ANN with a fixed number of epochs should result in a better model (smaller loss) on average with increasing sample size. Typically, such hyper-parameters can be substituted with other alternatives (e.g., a criterion that dynamically determines the number of epochs) so that performance is monotonic (on average) with sample size for any fixed values of the hyper-parameters. Thus, before using the above protocols an analyst is advised to consider whether the monotonicity assumption holds for all hyper-parameters.

Finally, we would like to comment on the use of Bayesian non-parametric techniques, such as Gaussian Processes.[34] Such methods consider and reason with all models of a given family, averaging out all model parameters to provide a prediction. However, they still have hyper-parameters. In this case, they are defined as the free parameters of the whole learning process over which there is no marginalization (averaging out). Examples of hyper-parameters include the type of the kernel covariance function in Gaussian Processes and the parameters of the kernel function.[35] In fact, since one can combine compositionally kernels via sum and product operations, dynamically composing the appropriate kernel adds a new level of complexity to hyper-parameter search.[36] Thus, in general, such methods still require hyper-parameters to tune and they do not completely

obviating the need to select them in our opinion. The protocols presented here could be employed to select these hyper-parameter values, type of kernel, type of priors, etc. From a different perspective however, the value of hyper-parameters in some settings (e.g., number of hidden units in a neural-network architecture) could be selected using a Bayesian non-parametric machinery. Thus, non-parametric methods could also substitute in some cases the need for the protocols in this paper.

## 10. Conclusions

In the absence of hyper-parameter optimization (model selection) simple Cross-Validation underestimates the performance of the model returned when training using the full dataset. In the presence of learning method and hyper-parameter optimization simple Cross-Validation overestimates performance. Some other alternatives are to rerun Cross-Validation one more time but only for the final selected model, a method proposed by Tibshirani and Tibshirani[4] to estimate and reduce the bias, and the Nested Cross Validation which Cross-Validates the model selection procedure (which includes an inner cross-validation). These alternatives seem to reduce bias with Nested Cross-Validation being conservative in general and robust to the dataset, although incurring a higher computational overhead; the TT method seems promising and does not require additional training of models. We would also like to acknowledge the limited scope of our experiments in terms of the number and type of datasets, the inclusion of other preprocessing steps into the analysis, the inclusion of other procedures for hyper-parameter optimization that dynamically decide to consider value combinations, using other performance metrics, experimentation with regression methods and others which form our future work on the subject in order to obtain more general answers to these research questions.

## Acknowledgments

The work was funded by the STATegra EU FP7 project, No. 306000, and by the EPILOGEAS GSRT ARISTEIA II project, No. 3446.

## References

1. D. Anguita, A. Ghio, L. Oneto and S. Ridella, In-sample and out-of-sample model selection and error estimation for support vector machines, *IEEE Trans. Neural Networks Learn. Syst.* **23**(9) (September 2012) 1390–1406.
2. G. C. Cawley and N. L. C. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, *J. Mach. Learn. Res.* **11** (March 2010) 2079–2107.
3. D. D. Jensen and P. R. Cohen, Multiple comparisons in induction algorithms, *Mach. Learn.* **38** (2000) 309–338.
4. R. J. Tibshirani and R. Tibshirani, A bias correction for the minimum error rate in cross-validation, *Ann. Appl. Stat.* **3**(2) (June 2009) 822–829.
5. R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in *Int. Joint Conf. on Artificial Intelligence* (1995), Vol. 14, pp. 1137–1143.
6. A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin and S. Levy, A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, *Bioinformatics* **21**(5) (March 2005) 631–643.

7. R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification* (October 2000), 2nd edn.
8. T. M. Mitchell, *Machine Learning* (March 1997).
9. C. M. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics) (August 2006).
10. T. Hastie, R. Tibshirani and J. Friedman, The elements of statistical learning, *Elements* **1** (2009) 337–387.
11. Y. Bengio and Y. Grandvalet, Bias in estimating the variance of K-fold cross-validation, in *Statistical Modeling and Analysis for Complex Data Problem* **1** (2005) 75–95.
12. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Series in Data Management Systems (June 2005), 2nd edn.
13. V. Lagani and I. Tsamardinos, Structure-based variable selection for survival data, *Bioinformatics* **26**(15) (2010) 1887–1894.
14. A. Statnikov, I. Tsamardinos, Y. Dosbayev and C. F. Aliferis, GEMS: A system for automated cancer diagnosis and biomarker discovery from microarray gene expression data, *Int. J. Med. Inform.* **74**(7–8) (August 2005) 491–503.
15. S. Salzberg, On comparing classifiers: Pitfalls to avoid and a recommended approach, *Data Min. Knowl. Discov.* **328** (1997) 317–328.
16. N. Iizuka, M. Oka, H. Yamada-Okabe, M. Nishida, Y. Maeda, N. Mori, T. Takao, T. Tamesa, A. Tangoku, H. Tabuchi, K. Hamada, H. Nakayama, H. Ishitsuka, T. Miyamoto, A. Hirabayashi, S. Uchimura and Y. Hamamoto, Oligonucleotide microarray for prediction of early intrahepatic recurrence of hepatocellular carcinoma after curative resection, *Lancet* **361**(9361) (March 2003) 923–929.
17. L. A. Kurgan, K. J. Cios, R. Tadeusiewicz, M. Ogiela and L. S. Goodenday, Knowledge discovery approach to automated cardiac SPECT diagnosis, *Artif. Intell. Med.* **23**(2) (October 2001) 149–169.
18. R. K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis and W. Wittek, Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope, *Nucl. Instruments Methods Phys. Res. Sect. A*: *Accel. Spectrometers, Detect. Assoc. Equip.* **516**(2–3) (January 2004) 511–528.
19. K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini and V. Consonni, Quantitative structure-activity relationship models for ready biodegradability of chemicals, *J. Chem. Inf. Model* **53** (2013) 867–878.
20. S. Moro and R. M. S. Laureano, Using data mining for bank direct marketing: An application of the CRISP-DM methodology, *Eur. Simul. Model. Conf.* (2011), pp. 117–121.
21. S. C. Bendall, E. F. Simonds, P. Qiu, E. D. Amir, P. O. Krutzik, R. Finck, R. V Bruggner, R. Melamed, A. Trejo, O. I. Ornatsky, R. S. Balderas, S. K. Plevritis, K. Sachs, D. Pe'er, S. D. Tanner and G. P. Nolan, Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum, *Science* **332**(6030) (May 2011) 687–696.
22. M. Sikora and L. Wrobel, Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines, *Arch. Min. Sci.* **55**(1) (2010) 91–114.
23. A. A. Aguilar-Arevalo, A. O. Bazarko, S. J. Brice, B. C. Brown, L. Bugel, J. Cao, L. Coney, J. M. Conrad, D. C. Cox, A. Curioni, Z. Djurcic, D. A. Finley, B. T. Fleming, R. Ford, F. G. Garcia, G. T. Garvey, C. Green, J. A. Green, T. L. Hart, E. Hawker, R. Imlay, R. A. Johnson, P. Kasper, T. Katori, T. Kobilarcik, I. Kourbanis, S. Koutsoliotas, E. M. Laird, J. M. Link, Y. Liu, Y. Liu, W. C. Louis, K. B. M. Mahn, W. Marsh, P. S. Martin, G. McGregor, W. Metcalf, P. D. Meyers, F. Mills, G. B. Mills, J. Monroe, C. D. Moore, R. H. Nelson, P. Nienaber, S. Ouedraogo, R. B. Patterson, D. Perevalov, C. C. Polly, E. Prebys, J. L. Raaf, H. Ray, B. P. Roe, A. D. Russell, V. Sandberg, R. Schirato, D. Schmitz, M. H. Shaevitz, F. C.

Shoemaker, D. Smith, M. Sorel, P. Spentzouris, I. Stancu, R. J. Stefanski, M. Sung, H. A. Tanaka, R. Tayloe, M. Tzanov, R. Van de Water, M. O. Wascko, D. H. White, M. J. Wilking, H. J. Yang, G. P. Zeller and E. D. Zimmerman, Search for electron neutrino appearance at the $\Delta m^2 \sim 1$ eV$^2$ scale, *Phys. Rev. Lett.* **98** (2007) 231801.

24. D. Coppersmith, S. J. Hong and J. R. M. Hosking, Partitioning nominal attributes in decision trees, *Data Min. Knowl. Discov.* **3** (1999) 197–217.
25. C.-C. Chang and C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.* **2**(27) (2011) 1–27.
26. I. Tsamardinos, V. Lagani and D. Pappas, Discovering multiple, equivalent biomarker signatures, in *7th Conference of the Hellenic Society for Computational Biology and Bioinformatics* (*HSCBB12*) (2012).
27. I. Tsamardinos, L. E. Brown and C. F. Aliferis, The max-min hill-climbing Bayesian network structure learning algorithm, *Mach. Learn.* **65**(1) (2006) 31–78.
28. T. Fawcett, An introduction to ROC analysis, *Pattern Recognit. Lett.* **27** (2006) 861–874.
29. A. Airola, T. Pahikkala, W. Waegeman, B. De Baets and T. Salakoski, A comparison of AUC estimators in small-sample studies, *J. Mach. Learn. Res. W&CP* **8** (2010) 3–13.
30. R. G. O'brien, A general ANOVA method for robust tests of additive models for variances, *J. Am. Stat. Assoc.* **74**(368) (December 1979) 877–880.
31. M. H. Quenouille, Approximate tests of correlation in time-series 3, *Math. Proc. Cambridge Philos. Soc.* **45**(03) (October 1949) 483–484.
32. S. Varma and R. Simon, Bias in error estimation when using cross-validation for model selection, *BMC Bioinformatics* **7** (January 2006) 91.
33. I. Tsamardinos, V. Lagani and A. Rakhshani, Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization, in *Proc. of the 8th Hellenic Conference on Artificial Intelligence* (*SETN'14*) (2014).
34. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (2006), Vol. 14.
35. T. Hofmann, B. Schölkopf and A. J. Smola, Kernel methods in machine learning, *The Annals of Statistics* **36**(3) (2008) 1171–1220.
36. D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum and Z. Ghahramani, Structure discovery in nonparametric regression through compositional kernel search, in *Proc. of the Int. Conf. on Machine Learning* (*ICML*) (2013), Vol. 30, pp. 1166–1174.

## Appendix A

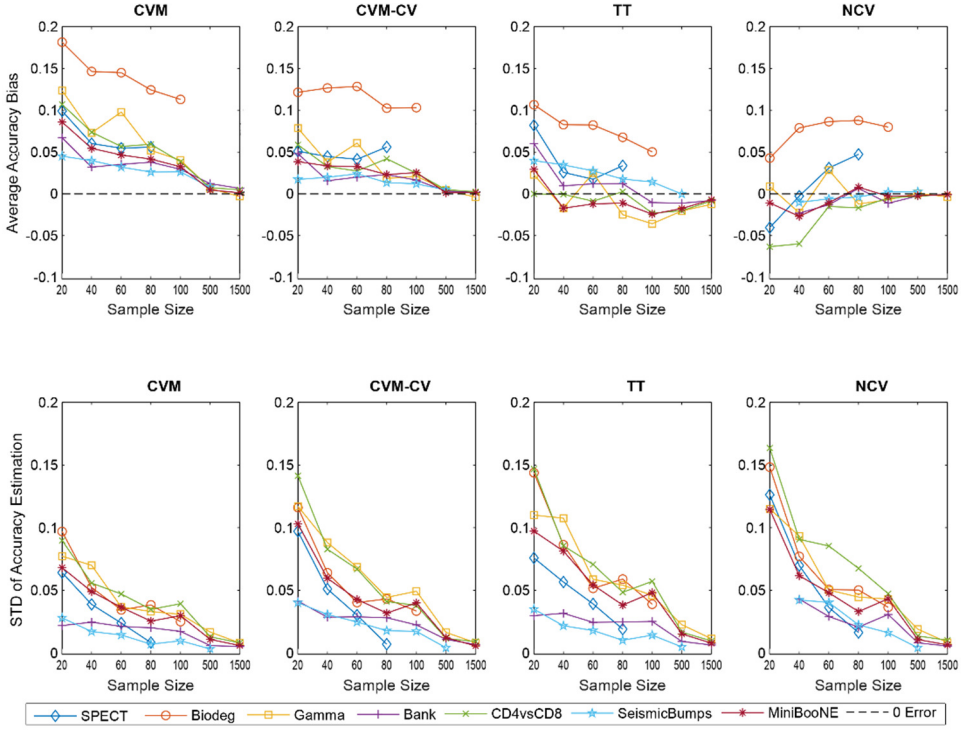


Fig. 4. (Color online) Average loss and variance for the accuracy metric in Strategy NFS. From left to right: stratified CVM, CVM-CV, TT, and NCV. Top row contains average bias, second row bias standard deviation. The results largely vary depending on the specific dataset. In general, CVM is clearly optimistic for sample sizes less or equal to 100, while NCV tends to underestimate performances. CVM-CV and TT show a behavior that is in between these two extremes. CVM has the lowest variance, at least for small sample sizes.

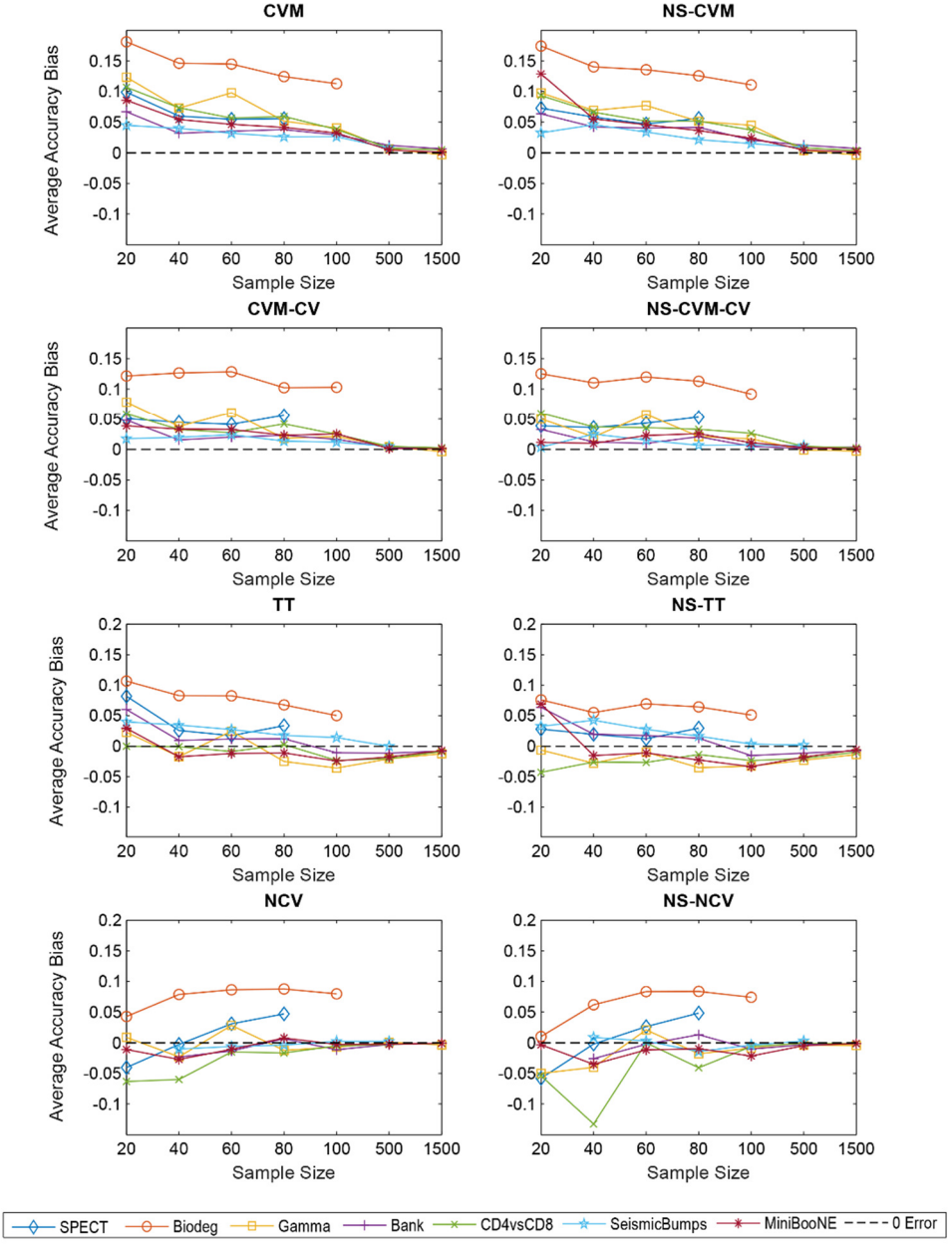


Fig. 5. (Color online) Effect of stratification using accuracy in Strategy NFS. The first column reports the average bias for the stratified versions of each method, while the second column for the non-stratified ones. The effect of stratification is dependent on each specific method and dataset.

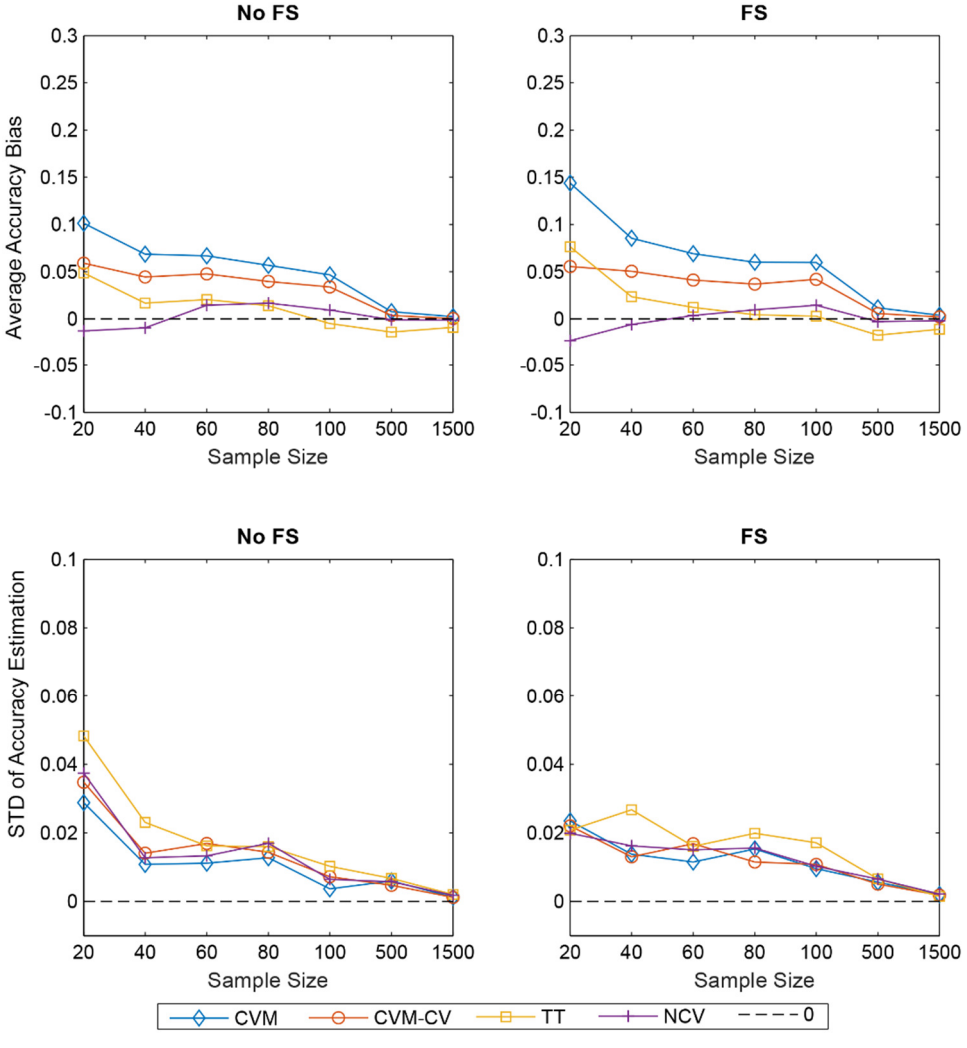


Fig. 6. (Color online) Comparing Strategy NFS (No Feature Selection) and WFS (With Feature Selection) over accuracy. The first row reports the average bias of each method for Strategy NFS and WFS, respectively, while the second row provides the bias standard deviation. CVM and TT shows an evident increment in bias in Strategy WFS, presumably due to the larger hyper-parameter space explored in this setting.

Table 6. Average Accuracy Bias over Datasets (Strategy *NFS*). *P*-values produced by a *t*-test with null hypothesis the mean bias is zero ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.1013** | 0.0948** | 0.0590** | 0.0458** | 0.0486** | 0.0315** | –0.0127 | –0.0311** |
| 40 | 0.0685** | 0.0682** | 0.0443** | 0.0357** | 0.0168** | 0.0096* | –0.0096 | –0.0236** |
| 60 | 0.0669** | 0.0617** | 0.0477** | 0.0434** | 0.0204** | 0.0113** | 0.0144** | 0.0171** |
| 80 | 0.0567** | 0.0549** | 0.0397** | 0.0389** | 0.0139** | 0.0074* | 0.0166** | 0.0090** |
| 100 | 0.0466** | 0.0422** | 0.0337** | 0.0263** | –0.0050 | –0.0084** | 0.0093 | 0.0039 |
| 500 | 0.0076** | 0.0075** | 0.0036** | 0.0028** | –0.0140** | –0.0141** | –0.0010 | –0.0020 |
| 1500 | 0.0023** | 0.0023** | 0.0004 | 0.0010 | –0.0091** | –0.0092** | –0.0015 | –0.0020* |

Table 7. Standard deviation of Accuracy estimations over Datasets (Strategy *NFS*). *P*-values produced by a test with null hypothesis that the variances are the same as the corresponding variance of the NCV protocol ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.0766** | 0.0862** | 0.1086** | 0.1067** | 0.1242 | 0.1509 | 0.1390 | 0.1374 |
| 40 | 0.0591** | 0.0600** | 0.0747* | 0.0826 | 0.0962 | 0.1033 | 0.0914 | 0.1118** |
| 60 | 0.0435** | 0.0463** | 0.0595 | 0.0537* | 0.0701 | 0.0786* | 0.0650 | 0.0639 |
| 80 | 0.0431** | 0.0454** | 0.0522* | 0.0520* | 0.0675 | 0.0732 | 0.0629 | 0.0709 |
| 100 | 0.0444** | 0.0422** | 0.0510 | 0.0508 | 0.0682** | 0.0663* | 0.0564 | 0.0575 |
| 500 | 0.0359 | 0.0357 | 0.0356 | 0.0366 | 0.0432* | 0.0436** | 0.0368 | 0.0370 |
| 1500 | 0.0278 | 0.0282 | 0.0281 | 0.0275 | 0.0290 | 0.0299 | 0.0277 | 0.0278 |

Table 8. Average Accuracy on the hold-out sets (Strategy *NFS*). All methods use CVM for model selection and thus have the same performances on the hold-out sets. NS stands for Non-Stratified.

| | CVM | NS-CVM |
|---|---|---|
| 20 | 0.7699 | 0.7639 |
| 40 | 0.8061 | 0.8016 |
| 60 | 0.8186 | 0.8203 |
| 80 | 0.8296 | 0.8280 |
| 100 | 0.8351 | 0.8377 |
| 500 | 0.8816 | 0.8814 |
| 1500 | 0.8805 | 0.8805 |

Table 9. Average AUC Bias over Datasets (Strategy *WFS*). *P*-values produced by a *t*-test with null hypothesis the mean bias is zero ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.2589** | 0.2821** | 0.0651** | 0.0128 | 0.2139** | 0.1487** | –0.0634** | –0.1124** |
| 40 | 0.1628** | 0.1729** | 0.0698** | 0.0418** | 0.0886** | 0.0691** | –0.0359* | –0.0418** |
| 60 | 0.1257** | 0.1349** | 0.0540** | 0.0362** | 0.051** | 0.0563* | –0.0081 | –0.0084 |
| 80 | 0.1034** | 0.1094** | 0.0475** | 0.0484** | 0.0321** | 0.0381 | –0.0222 | –0.017 |
| 100 | 0.0985** | 0.1029** | 0.0595** | 0.0525** | 0.0284** | 0.0294 | 0.0174** | 0.0017* |
| 500 | 0.0239** | 0.0259** | 0.0120** | 0.0143** | –0.0242** | –0.0014 | 0.0079 | 0.0018 |
| 1500 | 0.0007 | 0.0001 | –0.0007 | –0.0005 | –0.0122** | –0.0471** | –0.0031 | –0.0035** |

Table 10. Standard deviation of AUC estimations over Datasets (Strategy *WFS*). *P*-values produced by a test with null hypothesis that the variances are the same as the corresponding variance of the NCV protocol ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.0685** | 0.0596** | 0.2202 | 0.2390* | 0.1289** | 0.1303** | 0.2078 | 0.2456* |
| 40 | 0.0711** | 0.0769** | 0.1584** | 0.1911 | 0.1345** | 0.0974** | 0.2020 | 0.1896 |
| 60 | 0.0692** | 0.0669** | 0.1344 | 0.1590 | 0.1281* | 0.0859** | 0.1636 | 0.1669 |
| 80 | 0.0607** | 0.0624** | 0.1133** | 0.1214** | 0.1095** | 0.0773** | 0.1686 | 0.1581 |
| 100 | 0.0622** | 0.0643** | 0.1032* | 0.1283 | 0.1168 | 0.0717** | 0.1386 | 0.1689 |
| 500 | 0.0601** | 0.0599** | 0.0824 | 0.0740 | 0.1003** | 0.0468** | 0.0754 | 0.0882 |
| 1500 | 0.0408 | 0.0410 | 0.0424 | 0.0416 | 0.0519* | 0.0312** | 0.0443 | 0.0444 |

Table 11. Average AUC performance on the hold-out sets (Strategy *WFS*). All methods use CVM for model selection and thus have the same performances on the hold-out sets. NS stands for Non-Stratified.

| | CVM | NS-CVM |
|---|---|---|
| 20 | 0.6887 | 0.6871 |
| 40 | 0.7496 | 0.7471 |
| 60 | 0.7794 | 0.772 |
| 80 | 0.7983 | 0.7944 |
| 100 | 0.809 | 0.803 |
| 500 | 0.8643 | 0.863 |
| 1500 | 0.916 | 0.9165 |

Table 12. Average accuracy Bias over Datasets (Strategy *WFS*). *P*-values produced by a *t*-test with null hypothesis the mean bias is zero ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.1439** | 0.1507** | 0.0555** | 0.0214** | 0.0763** | 0.0757** | –0.0231 | –0.0612** |
| 40 | 0.0855** | 0.0854** | 0.0505** | 0.0395** | 0.0236** | 0.0185** | –0.0060 | –0.0325** |
| 60 | 0.0690** | 0.0671** | 0.0411** | 0.0393** | 0.0121* | 0.0075 | 0.0035 | 0.0039 |
| 80 | 0.0601** | 0.0583** | 0.0368** | 0.0395** | 0.0044 | 0.0026 | 0.0095* | –0.0013 |
| 100 | 0.0598** | 0.0584** | 0.0418** | 0.0378** | 0.0028 | 0.0005 | 0.0144** | 0.0132** |
| 500 | 0.0117** | 0.0103** | 0.0055** | 0.0045** | –0.0173** | –0.0192** | –0.0031 | –0.0035* |
| 1500 | 0.0038** | 0.0039** | 0.0021** | 0.0013 | –0.0109** | –0.0108** | –0.0020* | –0.0019* |

Table 13. Standard deviation of accuracy estimations over Datasets (Strategy *WFS*). *P*-values produced by a test with null hypothesis that the variances are the same as the corresponding variance of the NCV protocol ($P < 0.05$*, $P < 0.01$**). NS stands for Non-Stratified.

| | CVM | NS-CVM | CVM-CV | NS-CVM-CV | TT | NS-TT | NCV | NS-NCV |
|---|---|---|---|---|---|---|---|---|
| 20 | 0.0593** | 0.0703** | 0.1384 | 0.1407 | 0.1143 | 0.1313 | 0.1286 | 0.1482 |
| 40 | 0.0488** | 0.0528** | 0.0706 | 0.0734 | 0.0885 | 0.0948 | 0.0812 | 0.1028** |
| 60 | 0.0453** | 0.0472** | 0.0624* | 0.0612** | 0.0817 | 0.0858 | 0.0759 | 0.0711 |
| 80 | 0.0446** | 0.0440** | 0.0566 | 0.0500** | 0.0761* | 0.0766** | 0.0650 | 0.0776* |
| 100 | 0.0408** | 0.0420** | 0.0468** | 0.0484* | 0.0663* | 0.0703** | 0.0567 | 0.0566 |
| 500 | 0.0378 | 0.0371 | 0.0387 | 0.0388 | 0.0477* | 0.0469* | 0.0410 | 0.0405 |
| 1500 | 0.0299 | 0.0296 | 0.0300 | 0.0301 | 0.0318 | 0.0313 | 0.0300 | 0.0293 |

Table 14. Average accuracy performance on the hold-out sets (Strategy *WFS*). All methods use CVM for model selection and thus have the same performances on the hold-out sets. NS stands for Non-Stratified.

| | CVM | NS-CVM |
|---|---|---|
| 20 | 0.7571 | 0.7542 |
| 40 | 0.8051 | 0.8001 |
| 60 | 0.8206 | 0.8204 |
| 80 | 0.8308 | 0.8303 |
| 100 | 0.8333 | 0.8320 |
| 500 | 0.8791 | 0.8806 |
| 1500 | 0.8802 | 0.8804 |