Methodology and Statistics for the Behavioural, Biomedical and Social Sciences

MASTER'S THESIS

Evaluating tuning strategies for random forest hyperparameters with regards to prediction performance of clinical prediction models and computational time.

Judith Nève
0070661

Supervisors
Dr. Maarten van Smeden
Zoë Dunias

Word count
?/XXXX

1 Introduction

Machine learning models are increasingly used in medicine for diagnostic and prognostic purposes[1]. Risk estimates are used to assist clinical decisions (e.g., whether to undergo treatment). A popular set of techniques in clinical research is tree-based methods[2], among which random forests are particularly prevalent[3,4]. Random forests' results are influenced by hyperparameters (e.g., the number of predictors sampled to make a split)[5], which need to be set before training a model. Previous studies have shown classification accuracy (i.e., the proportion of observations correctly classified) to have little-to-no improvement when using optimal hyperparameters for a given dataset compared to hyperparameter software defaults[6,7]. Classification accuracy is however not sufficient to evaluate a model's clinical utility: identifying a patient as positive or negative may not carry sufficient information, as the patient's predicted risk is a key element of medical decision-making. Moreover, classification accuracy depends on the chosen classification threshold, i.e., the predicted risk value above which an observation is classified as positive. For clinical purposes, discrimination (i.e., positive patients have higher risk predictions than negative patients) and calibration (i.e., risk predictions reflect patients' true risk)[8] are two much more important performance metrics. Models with low discrimination may misclassify patients, while miscalibrated models can lead to over- or undertreatment if over- or underestimating patients' risks. Random forests can heavily suffer from miscalibration[9,10].

A well-known way to improve model performance is to perform hyperparameter tuning, which identifies hyperparameter values for which a performance metric (e.g., deviance) is optimal for a given prediction model. This procedure requires choosing which metric is optimised, which hyperparameters are tuned, and how candidate hyperparameter values are generated. The latter two can greatly impact computational time: generally, the more hyperparameters are tuned or the more candidate values there are, the larger computational time is. Hyperparameter search algorithms may consider all possible hyperparameter values and their combinations, or use information collected earlier in the tuning procedure to tune efficiently. Computational time is important to consider in conjunction to the improvement in model performance: greater computational times and longer training periods of models may lead to a strain on resources and increase carbon emissions. For complex models, this can be as much as five times the average emissions of a car over its lifetime[11].

There has been little focus on comparing tuning procedures on clinically relevant performance metrics. Previous research suggests the most gain in random forest performance as measured by classification accuracy, discrimination, and the Brier score could be made by tuning the number of predictors sampled to make a split and the proportion of the sample used to fit the tree[12]. Yang and Shami[13] found tuning using most hyperparameter search algorithms improved classification accuracy compared to default hyperparameters. Some search algorithms had greatly increased computational times without improving classification accuracy further than shorter search algorithms. So far, no study has compared different optimisation metrics' effects on model performance or computational time. Moreover, existing studies focusing on tuning procedures for random forests use high-dimensional datasets (datasets with more features than observations) whereas clinical research typically uses low-dimensional datasets (datasets with more observations than features), for which guidance regarding hyperparameter tuning is lacking[14].

The current project addresses the following question: what are the optimal hyperparameter tuning strategies for balancing predictive performance and computational time of tree-based methods in low-dimensional settings? There are three simulation studies in this project to identify (i) which hyperparameters to tune, (ii) which metric to optimise, and (iii) which search algorithm to use for optimal model performance.

2 Methods

This method section follows the ADEMP approach[15].

2.1 Aim

2.1.1 Study 1: Hyperparameters to tune

Using datasets from the OpenML platform[16], Probst and colleagues[12] found the number of predictors considered at a split and the sample fraction to be the two most influential hyperparameters on the discriminative performance of prediction models. These findings only investigated the effect of tuning one or two hyperparameters at once. Our first study aimed to extend these findings by considering (i) model calibration in addition to model discrimination and (ii) more combinations of hyperparameters. We examined the effect of tuning different combinations of hyperparameters on the performance of a prediction model and evaluate performance improvements in context of required computational times.

2.1.2 Study 2: Optimisation metric

text here.

2.1.3 Study 3: Hyperparameter search algorithm

text here.

2.2 Data-generating mechanism

All studies used the same data-generating mechanism. Different datasets were generated for each study.

2.2.1 Data-generating scenarios

A full factorial simulation design was used to consider the influence of data characteristics on model predictive performance and computational time. The varying simulation factors were the number of candidate predictors p (range: 8, 16), the event fraction EF (range: 0.1, 0.3, 0.5), and the sample size N (range: $0.5n, n$). n is the minimum sample size required to identify effects for a given number of regression coefficients (here, $1.25p$), expected event fraction, and expected AUC (here, 0.8), as calculated by Riley and colleagues[17]. A total of 12 (2*3*2) scenarios were considered. For each scenario, 500 training datasets were generated, yielding a total of 6,000 datasets per study.

Training and validation datasets were simulated under a logistic model with strong interactions. For each observation i ($i = 1, \ldots, N$), predictors $\mathbf{x}_i$ were drawn from a p-variate normal distribution with parameters detailed in Formula 1. $0.25p$ two-way interactions were included, with the h^{th} ($h = 1, ..., 0.25p$) interaction being the product of the h^{th} and the $(h + p/4)^{th}$ predictors. The binary outcome y_i was drawn from a Bernoulli distribution conditional on the predictor values (Formula 2).

$$\mathbf{x}_i \sim \text{MVN}(\mathbf{0}, \begin{bmatrix} 1 & 0.2 & ... \\ 0.2 & 1 & ... \\ ... & ... & ... \end{bmatrix}). \tag{1}$$

$$P(y_i = 1|\mathbf{x}_i) = \frac{1}{1 + exp(-(\beta_0 + \sum_{j=1}^{p/2} \beta x_{ij} + \sum_{j=1+p/2}^{3p/4} 2\beta x_{ij} + \sum_{h=1}^{0.25*p} \gamma x_{ih} x_{i(h+0.25p)}))}. \tag{2}$$

In Formula 2, β_0, β, γ are respectively the intercept, main effect regression coefficient, and interaction effect regression coefficient of the data generating model, hereafter called "true effects".

A validation dataset ($N = 100,000$) was generated for each training dataset in order to evaluate model performance.

2.2.2 True effect estimation

True effects were determined for each unique combination of p and EF. Let k denote a given combination. For each combination, the intercept $\beta_0^{(k)}$, predictor main effects $\boldsymbol{\beta}^{(k)}$, and predictor interaction effects $\boldsymbol{\gamma}^{(k)}$ were estimated using a large sample approximation ($N = 100,000$). The first $p/2$ main effects were set to be equal (i.e., $\beta_1^{(k)} = \beta_2^{(k)} = ... = \beta_{p/2}^{(k)}$). The next $p/4$ main effects were set to be twice the strength of the first $p/2$ main effects (i.e., $\beta_{1+p/2}^{(k)} = \beta_{2+p/2}^{(k)} = ... = \beta_{3p/4}^{(k)} = 2\beta_1^{(k)}$). The final $p/4$ main effects were set to 0. All interaction effects $\boldsymbol{\gamma}^{(k)}$ were set to be equal (i.e., $\gamma_1^{(k)} = \gamma_2^{(k)} = ... = \gamma_{0.25p}^{(k)}$). The R function `optim` was used to minimise a loss function measuring the sum of the squared difference between (i) 0.7 and the observed AUC in a model with no interactions, (ii) 0.8 and the observed AUC in a model with interactions, and (iii) the targeted event fraction and the average estimated probability $P(y_i = 1|\mathbf{x}_i)$ in the simulated dataset. This was repeated 20 times for each scenario. All generated effects were used to generate a validation dataset of $N = 100,000$. For each scenario, the best-performing set of effects was selected to be the set of effects with the smallest sum of absolute differences between (i) target prevalence and observed prevalence, (ii) 0.7 and the AUC of a model with no interactions, and (iii) 0.8 and the AUC of a model with interactions. A test dataset was generated to obtain performance estimates for the set of true effects. True effects and their performance are presented in Table 1.

2.3 Methods

2.3.1 Study 1: Hyperparameters to tune

We varied which hyperparameters were tuned when fitting a random forest model using the R package `ranger`[18] via the R package `caret`[19]. We used grid search to optimise deviance. 5-fold cross-validation was used.

Considering all possible combinations of 5 hyperparameters leads to 32 tuning procedures on each dataset. Due to the unfeasibility of such a large-scale approach, we reduced the number of combinations studied: only hyperparameter combinations including at least `mtry` (the number of predictors randomly sampled to make a split) and `min.node.size` (the minimum number of observations for a node to be formed, i.e., the point at which a split should not be computed regardless of impurity) were considered. Other hyperparameters which could be included in combinations were `replace` (whether the data used to fit a single tree is sampled with or without replacement), `sample.fraction` (the proportion of the data used to fit a single tree), and `splitrule` (the way in which a split is picked). Default values and tuning ranges are presented in Table 2.

This yielded 8 hyperparameter combinations. Hyperparameters not included in a given combination were set to their default value. A model using the default hyperparameters was fit to establish the baseline. All

Table 1: Performance of selected true effects

p	EF (target)	β_0	β	γ	AUC (no interaction)	AUC (interaction)	EF (obtained)
8	0.10	-2.98	0.14	0.66	0.71	0.80	0.10
8	0.30	-0.94	0.24	-0.84	0.70	0.80	0.30
8	0.50	-0.22	0.22	0.83	0.70	0.80	0.50
16	0.10	-3.06	0.07	0.44	0.70	0.80	0.10
16	0.30	-0.85	0.14	-0.55	0.70	0.80	0.30
16	0.50	-0.31	-0.13	0.55	0.70	0.80	0.50

Table 2: Hyperparameter tuning ranges

Hyperparameter	Default	Range
`mtry`	$\sqrt{p}$ (rounded down)	1-p
`min.node.size`	1	1-10
`replace`	TRUE	TRUE, FALSE
`sample.fraction`	1	0.1, 0.2, ..., 0.9, 1
`splitrule`	gini	gini, hellinger, extratrees

For `splitrule` = extratrees, an additional parameter should be considered regarding the number of random splits to consider. This was set to its default of 1.

considered combinations were used to fit a random forest on each training dataset, leading to 90 tuning procedures being performed for these preliminary results.

2.3.2 Study 2: Optimisation metric

text here.

2.3.3 Study 3: Hyperparameter search algorithm

text here.

2.4 Performance measures

The same outcomes were used in all studies. Primary outcomes were defined as those which were used to advise on the best tuning procedures. These measures were discrimination, calibration slope, root mean square of the log of the calibration slope over all the runs of a scenario (RMSD(slope)), and computational time. Discrimination was measured by the AUC and assessed whether positive observations have higher predicted risk than negative observation. Calibration was measured by the calibration slope (as calculated by Van Calster[20]) and assessed the extent to which predicted risks reflect true risks. RMSD(slope) assessed the extent to which the calibration slopes varied within a data-generating mechanism and how much they differed from the ideal value of 1, as used by Van Calster and colleagues[21]. Computational time was the amount of time for which a given tuning procedure ran, measured in seconds.

Candidate optimisation metrics from study 2 were also included as secondary outcomes in all studies. These were the classification accuracy (the proportion of correctly classified observations at a threshold of 0.5) the calibration intercept (the extent to which the mean predicted risk reflects the true prevalence), the Brier score (which combines calibration and discrimination components), the logarithmic loss (how close the predicted risk comes to the observed outcome), and Cohen's Kappa (the proportion of correctly classified observations, accounting for chance).

Model performance metrics were estimated using the predictions of the model on an independently generated validation set generated under the same data generating mechanisms. For each data simulation scenario and tuning procedure combination, we computed the average and Monte Carlo error of each of these performance measures.

We evaluated and compared model predictive performance between hyperparameter combinations. We visualized whether certain hyperparameter combinations had a notably larger runtime compared to others, for a relatively low increase in performance. The best hyperparameter combination was assessed considering all primary outcomes.

2.5 Software

Data was simulated and tuning procedures were performed on a high-performance computer. This was done in R version 4.2.2. Summary statistics and figures were generated on a personal computer using R version 4.1.2[22].

3 Results

3.1 Study 1: Hyperparameters to tune

The average and Monte Carlo error of primary outcomes for each hyperparameter combination are presented in Table 3. Effects were homogeneous across scenarios. We illustrate results specific to the scenario where $p = 8$, $EF = 0.3$, and $N = n$, in Figure 1. Results for other scenarios can be visualised at [THIS WILL BE A LINK TO A SHINY APP].

These results show varying hyperparameter tuning combinations had very limited effects on discrimination, with a mean AUC of 0.7 (SD = 0.02) across all hyperparameter combinations. A large effect was observed on model calibration and computational time. Model calibration varied considerably within and between combinations (Table 3, Figure 1). The best calibration performance was obtained by tuning `mtry` and `min.node.size`, which had a median calibration slope of 1.07 across all scenarios. This performance was among the most stable with an IQR of 0.29. [if I do make calibration plots, comments on them can go here]. Tuning `mtry`, `min.node.size`, and `replace` performed similarly well, but took twice as long to run. `mtry` and `min.node.size` was thus selected as the best combination to tune.

More generally, large increases in runtime were observed when tuning 4 or more hyperparameters, with noticeably worse calibration performance and no observable change in other performance measures (Figure 1).

Secondary outcomes showed similar patterns (Figure 1). Classification accuracy, Brier score, and logarithmic loss showed very little variation both within and between hyperparameter combinations. Calibration intercept and Cohen's Kappa showed variations within, but not between, hyperparameter combinations.

4 Discussion

DON'T READ ON

In this report, we studied the impact of the choice of hyperparameters being tuned on model performance and computational time. This was a first step in our aim to identify optimal hyperparameter tuning strategies for balancing predictive performance and computational time of tree-based methods for the development of clinical prediction models in low-dimensional settings. We presented preliminary results computed using one

Table 3: Average performance of hyperparameter combinations (aggregated over all scenarios)

Tuned hyperparameters	AUC Mean (SD)	Calibration slope Median (IQR)	RMSD(slope)	Runtime (seconds) Mean(SD)
None	0.7 (0.02)	0.87 (0.25)	0.25	1.4 (0001.0)
mtry + min.node.size	0.7 (0.02)	1.07 (0.29)	0.20	172.9 (0232.3)
mtry + min.node.size + replace	0.7 (0.02)	1.06 (0.29)	0.21	394.5 (0540.2)
mtry + min.node.size + splitrule	0.71 (0.02)	1.56 (0.58)	0.49	486.1 (0581.8)
mtry + min.node.size + replace + splitrule	0.71 (0.02)	1.55 (0.59)	0.49	1114.0 (1354.7)
mtry + min.node.size + sample.fraction	0.7 (0.02)	1.44 (0.49)	0.43	1130.8 (1399.7)
mtry + min.node.size + sample.fraction + replace	0.7 (0.02)	1.43 (0.5)	0.44	2479.7 (3136.8)
mtry + min.node.size + sample.fraction + splitrule	0.7 (0.02)	1.98 (1.22)	0.77	3199.0 (3531.2)
mtry + min.node.size + sample.fraction + replace + splitrule	0.7 (0.02)	1.97 (1.29)	0.77	7063.2 (8003.0)

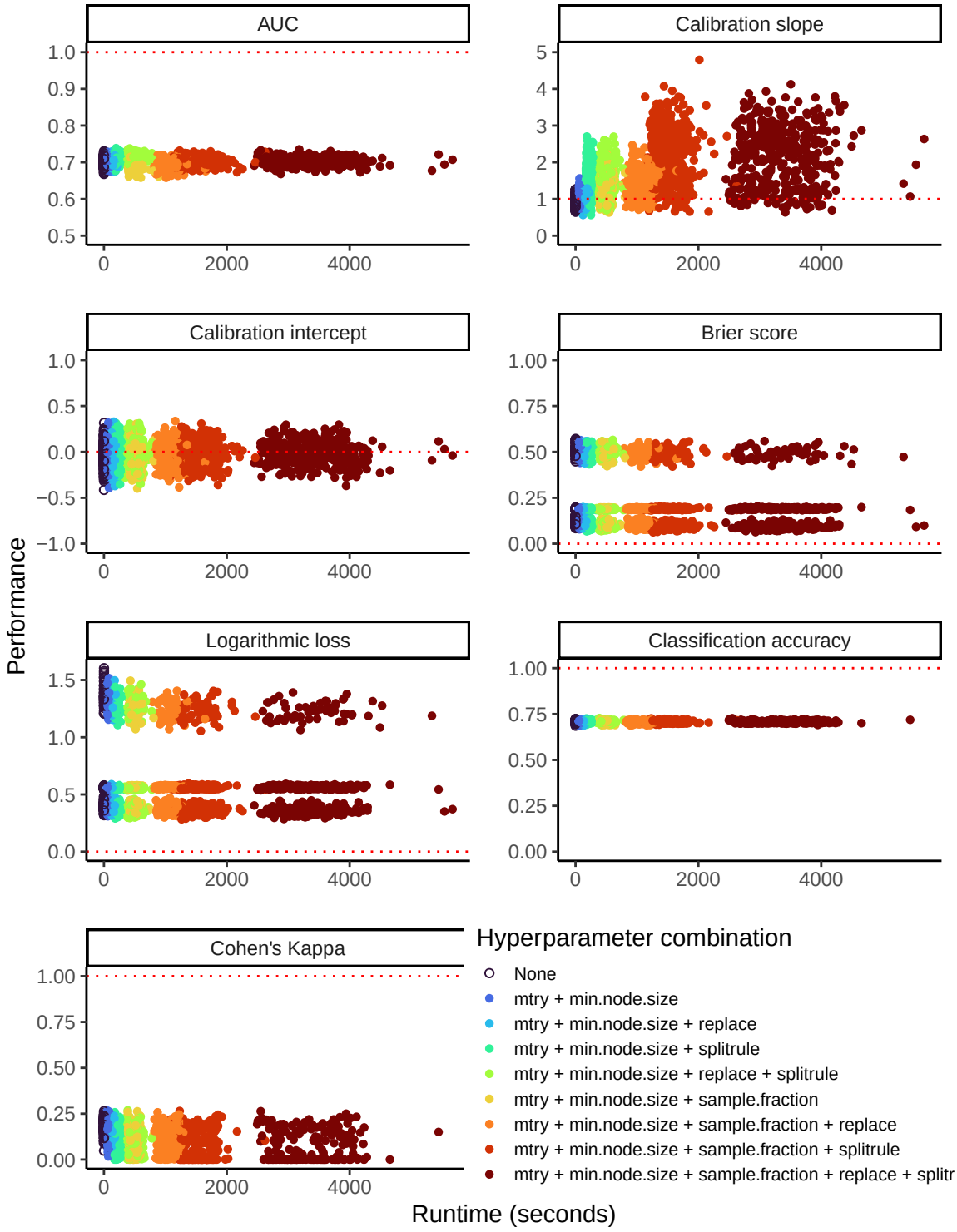

Figure 1: Performance of each hyperparameter tuning combination on the example scenario where p = 8, EF = 0.3, N = n

data simulation scenario and ten simulation runs. Tuning led to increases in runtime with large but heterogeneous effects on model calibration. Other performance measures showed little-to-no effects. Computational time was highly influenced by hyperparameter combinations: in this simple scenario, this ranged from 0.2 seconds to over 40 minutes. These preliminary results suggest tuning number of predictors, replacement, minimum node size, and splitting rule leads on average to the best model performance. This hyperparameter combination had large amounts of variations. Not tuning hyperparameters led to more stable calibration performance, but consistently overestimated risk.

Findings regarding the impact of hyperparameter combination on computational time match expectations: tuning more hyperparameters increased computational time. This may have been exacerbated by using grid search to search for the optimal hyperparameters, as it is the most computationally intensive hyperparameter candidate value generation algorithm[13].

While these results are only preliminary and observed effects on model predictive performance may thus be a product of chance, it is important to consider alternative explanations for the observed results. Tuning is not beneficial for all data[7]: default hyperparameters may be close to optimal for the presented data-generating mechanism. This will be explored in the full-scale study, which will consider various event fractions, sample sizes, and numbers of predictors. We may also examine scenarios in which predictors have unequal effects, as one of the advantages of random forests is their ability to include variables with smaller, yet relevant, effects[6,7]. Alternatively, optimal hyperparameter values may not have been identified. As classification accuracy is optimised, this may lead to missing optimal values for other performance metrics. In grid search, candidate values for each hyperparameter are determined by the user, leading to possibly missing the optimal value in low-dimensional settings[21,23]. This highlights possible large systematic problems in random forest tuning procedures, which are heavily focused on obtaining good classification accuracy in high-dimensional settings. We will investigate how predictive model performance can be improved by varying optimisation metrics and candidate hyperparameter search algorithms to adapt tuning procedures to low-dimensional, clinical settings.

Random forests are widely used in clinical research[3,4], yet can suffer from miscalibration[9,10]. In this preliminary study, we examined the impact of tuning different sets of hyperparameters on the predictive performance of random forest models for dichotomous risk prediction and on computational time. Results suggested tuning may have large effects on model predictive performance and computational time. This study will be conducted on a larger scale to provide more reliable findings, and follow-up studies will be conducted to investigate the effect of optimisation metrics and hyperparameter candidate value generation algorithms. We aim to provide guidelines regarding possible trade-offs between computationally intensive tuning procedures and model predictive performance. While this project has so far focused on random forests, other tree-based methods may be explored in future.

References

1. Wessler, B. S. *et al.* Tufts PACE clinical predictive model registry: update 1990 through 2015. *Diagnostic and prognostic research* **1,** 1–8 (2017).
2. Mitchell, T. M. & Mitchell, T. M. *Machine learning* (1997).
3. Uddin, S., Khan, A., Hossain, M. E. & Moni, M. A. Comparing different supervised machine learning algorithms for disease prediction. en. *BMC Medical Informatics and Decision Making* **19,** 281 (2019).
4. Andaur Navarro, C. L. *et al.* Systematic Review Identifies the Design and Methodological Conduct of Studies on Machine Learning-Based Prediction Models. *Journal of Clinical Epidemiology* **154,** 8–22 (2022).
5. Mantovani, R. G. *et al.* An empirical study on hyperparameter tuning of decision trees. http://arxiv.org/abs/1812.02207 (2019).
6. Bernard, S., Heutte, L. & Adam, S. *Influence of Hyperparameters on Random Forest Accuracy* in *Multiple Classifier Systems* (2009), 171–180.
7. Probst, P., Wright, M. N. & Boulesteix, A.-L. Hyperparameters and tuning strategies for random forest. en. *WIREs Data Mining and Knowledge Discovery* **9,** e1301 (2019).
8. Van Calster, B. *et al.* Calibration: the Achilles heel of predictive analytics. *BMC Medicine* **17,** 230 (2019).
9. Benedetto, U. *et al.* Can Machine Learning Improve Mortality Prediction Following Cardiac Surgery? *European Journal of Cardio-Thoracic Surgery* **58,** 1130–1136 (2020).
10. Djulbegovic, B., Berano Teh, J., Wong, L., Hozo, I. & Armenian, S. H. Diagnostic Predictive Model for Diagnosis of Heart Failure after Hematopoietic Cell Transplantation (HCT): Comparison of Traditional Statistical with Machine Learning Modeling. *Blood* **134,** 5799 (2019).
11. Strubell, E., Ganesh, A. & McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. http://arxiv.org/abs/1906.02243 (2019).
12. Probst, P., Boulesteix, A.-L. & Bischl, B. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research* **20,** 1934–1965 (2019).
13. Yang, L. & Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. en. *Neurocomputing* **415,** 295–316 (2020).
14. Ellenbach, N., Boulesteix, A.-L., Bischl, B., Unger, K. & Hornung, R. Improved Outcome Prediction Across Data Sources Through Robust Parameter Tuning. en. *Journal of Classification* **38,** 212–231 (2021).
15. Morris, T. P., White, I. R. & Crowther, M. J. Using simulation studies to evaluate statistical methods. en. *Statistics in Medicine* **38,** 2074–2102 (2019).
16. Bischl, B. *et al.* OpenML Benchmarking Suites. arXiv: 1708.03731. http://arxiv.org/abs/1708.03731 (2021).
17. Riley, R. D. *et al.* Calculating the sample size required for developing a clinical prediction model. *BMJ* **368,** 441 (2020).
18. Wright, M. N. & Ziegler, A. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software* **77,** 1–17 (2017).
19. Kuhn, M. *caret: Classification and Regression Training* R package version 6.0-90 (2021).
20. Van Calster, B. *benvancalster/classimb_calibration* 2022. https : / / github . com / benvancalster / classimb _ calibration / blob / ad521b46b32ec42689a05bc336fb3270a5c1f28e / simulation % 20study / Simulation/performance_measures_wo_eci.R.
21. Van Calster, B., van Smeden, M., De Cock, B. & Steyerberg, E. W. Regression shrinkage methods for clinical prediction models do not guarantee improved performance: Simulation study. *Statistical Methods in Medical Research* **29,** 3166–3178 (2020).

22. R Core Team. *R: A Language and Environment for Statistical Computing* version 4.1.2. R Foundation for Statistical Computing (2021).

23. Van Smeden, M. *et al.* Sample Size for Binary Logistic Prediction Models: Beyond Events per Variable Criteria. *Statistical Methods in Medical Research* **28,** 2455–2474 (2019).