Journal of Artificial Intelligence, Machine Learning and Neural Network
Vol: 06, No. 02, July-Dec 2026
ISSN: 2799-1172, DOI: 10.55529/jaimlnn.62.120.133

Research Paper

Tuned on paper, not in practice: How much does hyperparameter tuning improve classifiers on small tabular datasets?

Parvinder S. Sandhu[1*], Jhosep Sánchez-Flores[2]
[1*]Professor and Director-Principal (Ex Rayat-Bahra Institute of Engineering & Bio-Technology, Mohali, Punjab, India.
[2]Escuela Profesional de Ingeniería de Sistemas e Informática, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional de San Martín, Tarapoto, Peru.

Article Info

Article History:
Received: 01 July 2026
Revised: 30 August 2026
Accepted: 19 September 2026
Published: 22 October 2026

Keywords:
Hyperparameter Optimisation
Nested Cross-Validation
Random Search
Selection Bias
Default Hyperparameters
Tabular Data

ABSTRACT

Hyperparameter tuning is routinely reported as a source of improvement in applied machine learning, but the gain is usually judged by the same cross-validation scores that were used to select the hyperparameters. This study measures how much of that gain survives on data untouched by the search. Logistic regression, a radial-basis support vector machine, a random forest, histogram gradient boosting and a multilayer perceptron were evaluated on seven public tabular datasets with 208–1797 samples, comparing library default hyperparameters with configurations found by random search over 20 candidates. A nested design was used: tuning took place inside 3-fold cross-validation on the development data of each of five outer folds, and both the defaults and the selected configurations were then scored once on the held-out outer fold, giving 175 paired comparisons. Measured by the inner cross-validation, tuning improved balanced accuracy by 1.34 percentage points on average and appeared to beat the defaults in 66–100% of runs, depending on the model. On the held-out folds, the realised gain was 0.34 percentage points, tuning improved accuracy in only 76 of 175 comparisons (30 were unchanged), and no model showed a significant gain (Wilcoxon $p \geq 0.455$). The apparent gain was therefore about four times the realised gain ($p = 2.2 \times 10{-}5$). The discrepancy arose because the inner score of the selected configuration was optimistically biased (+0.26 points) while that of the defaults was pessimistically biased (−0.74 points). Increasing the search budget from 5 to 20 trials did not increase the realised gain consistently, and the best tuned model outperformed the best default model on only two of seven datasets. On small tabular datasets, defaults are strong baselines, and reported tuning gains should be measured on held-out data rather than on the scores used for selection.

Corresponding Author:

Journal of Artificial Intelligence, Machine Learning and Neural Network (JAIMLNN)

Parvinder S. Sandhu
Professor and Director-Principal (Ex Rayat-Bahra Institute of Engineering & Bio-Technology, Mohali, Punjab, India.
Email: Parvsandhu73@gmail.com

Copyright © 2026 The Author(s). This is an open access article distributed under the Creative Commons Attribution License, (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Almost every applied machine learning (ML) study reports that its models were "tuned". Hyperparameter optimisation (HPO) has developed into a field of its own, with random search [1], sequential model-based methods [2], [3], bandit-based schemes [4] and comprehensive reviews [5], [6]. Tuning is widely regarded as essential for a fair comparison of learning algorithms, and apparent improvements over untuned baselines are a common part of the evidence offered for new methods [7], [8].

Two observations suggest that the benefit of tuning on small datasets may be smaller than it appears. First, large-scale studies of tunability found that, for many algorithms and hyperparameters, well-chosen defaults are close to the best achievable settings [9], [10], [11]. Second, the scores used to choose a configuration are biased estimates of its performance: selecting the best of many noisy cross-validation estimates inflates the winner's score, a selection bias documented by Varma and Simon [12] and Cawley and Talbot [13]. If the improvement from tuning is reported on the same scores used to select the configuration, part of it may be an artefact of the selection itself.

This paper quantifies that artefact for five standard classifiers on seven small public tabular datasets. Using nested cross-validation [12], [14], it compares library default hyperparameters with configurations found by random search and measures both the apparent gain, as seen by the inner cross-validation that drives the search, and the realised gain on held-out data. The contributions are as follows:

- A paired comparison of tuned and default configurations over 175 held-out folds (seven datasets, five models and five outer folds), showing that the realised gain from random search averaged 0.34 percentage points of balanced accuracy and was not significant for any model.
- Evidence that the apparent gain measured by the inner cross-validation was about four times larger than the realised gain, and a decomposition of this discrepancy into the optimism of the selected configuration and the pessimism of the default's inner score.
- An analysis of search budget and of the best achievable model per dataset, showing that larger budgets did not reliably help and that tuning changed the best model only rarely.
- Practical recommendations for reporting tuning gains in applied ML studies.

2. RELATED WORK

2.1 Hyperparameter Optimisation

Bergstra and Bengio [1] showed that random search is more efficient than grid search when only a few hyperparameters matter, because it explores more distinct values of each. Model-based methods such as the tree-structured Parzen estimator [2] and Gaussian-process Bayesian optimisation [3] use earlier evaluations to guide the search, and Hyperband [4] allocates resources adaptively to promising configurations. Feurer and Hutter [5] and [6] review these methods and emphasise that the evaluation protocol is as important as the search algorithm. In this study, random search was chosen deliberately: it is the most widely used method in applied work, it is simple to reproduce, and it is a strong baseline for low-dimensional search spaces [1].

2.2 Tunability and Default Hyperparameters

[9] defined the tunability of an algorithm as the expected improvement of tuned over default settings and estimated it on 38 datasets from OpenML; they found considerable differences between algorithms and hyperparameters, with some defaults already close to optimal. Van Rijn and Hutter [10] studied which hyperparameters matter most across datasets, and [11] found that for many algorithms tuning offered small improvements over defaults. Large comparisons of classifiers on tabular data [15], [16], [17] show that differences between well-configured models are often small relative to their variability, which makes the size of tuning gains, and the reliability with which they are measured, directly relevant to algorithm comparisons.

2.3 Selection Bias in Model Evaluation

Stone [14] introduced cross-validatory assessment of predictive models. Varma and Simon [12] showed that reporting the cross-validation error of a configuration selected by that same cross-validation is optimistically biased and that nested cross-validation removes the bias. Cawley and Talbot [13] demonstrated that over-fitting of the model-selection criterion can be as damaging as over-fitting of the model parameters, particularly with small samples. More recent work on benchmarking has stressed that variability between runs [18] and the budget spent on tuning [7], [19] must be reported to make comparisons meaningful. The present study applies these ideas to the specific question of how large the gain from tuning is, as opposed to how accurate the tuned model is.

3. METHODOLOGY

3.1 Datasets

Seven public classification datasets with 208 to 1797 samples were used Table 1: the Wisconsin diagnostic breast cancer data (WDBC) [20], the Pima Indians diabetes data [21], the sonar mines-versus-rocks data [22], the ionosphere radar data [23], the Cleveland heart disease data [24], the wheat seeds data [25] and the handwritten digits data [26]. In the Pima data, physiologically impossible zero values of five clinical measurements were treated as missing; in the Cleveland data, six incomplete records were removed and two categorical variables were one-hot encoded. All other features were used as provided. Missing values were imputed with the training-fold median inside each model pipeline.

Table 1. Datasets Used in the Study

Dataset	Samples	Features	Classes	Class Sizes	Domain
WDBC [20]	569	30	2	212 / 357	Medical imaging
Pima [21]	768	8	2	500 / 268	Clinical
Sonar [22]	208	60	2	97 / 111	Signal processing
Ionosphere [23]	351	34	2	126 / 225	Radar
Cleveland [24]	297	16	2	160 / 137	Clinical
Seeds [25]	210	7	3	70 / 70 / 70	Agriculture
Digits [26]	1797	64	10	174–183 each	Image recognition

3.2 Models, Defaults and Search Spaces

Five classifiers were implemented in scikit-learn [27]: L2-regularised logistic regression (LR) [28], a support vector machine with radial-basis kernel (SVM) [29], a random forest (RF) with 150 trees [30], histogram gradient boosting (GB) [31], [32] and a single-hidden-layer multilayer perceptron (MLP) [33] trained with Adam [34]. LR, SVM and MLP inputs were standardised within each training fold. The default configuration of each model was the scikit-learn default. The search spaces Table 2 covered the hyperparameters most commonly tuned for each model, with log-uniform sampling for scale parameters, and were chosen to contain the defaults.

Table 2. Default Values and Random-Search Spaces. U = Uniform; LU = Log-Uniform; I = Integer Uniform

Model	Hyperparameter	Default	Search Space
LR	Inverse regularisation C	1.0	LU(10−4, 104)
SVM	C; kernel width γ	1.0; 1/(p·Var(X))	LU(10−2, 103); LU(10−4, 10)
RF	Fraction of features per split; min. samples per leaf	√p / p; 1	U(0.05, 1.0); I(1, 20)
GB	Learning rate; iterations; max. leaves; min. samples per leaf; L2 penalty	0.1; 100; 31; 20; 0	LU(0.01, 0.3); I(50, 300); I(4, 64); I(5, 50); LU(10−6, 1)
MLP	Hidden units; L2 penalty α; initial learning rate	100; 10−4; 10−3	I(16, 256); LU(10−6, 0.1); LU(10−4, 0.01)

3.3 Nested Evaluation Protocol

The protocol is illustrated in Figure 1. Each dataset was split into five stratified outer folds. For each outer split, the four development folds were used for tuning and the fifth, untouched fold for testing. Within the development data, 3-fold stratified cross-validation was used to score the default configuration and 20 configurations drawn at random from the search space. The configuration with the highest mean inner score after the first 5, 10 and 20 trials was refitted on all development data, as was the default, and each was scored once on the outer test fold. The performance measure throughout was balanced accuracy, the mean recall over classes, which is appropriate for the imbalanced and multiclass datasets. The procedure yielded 35 paired comparisons per model and 175 in total.

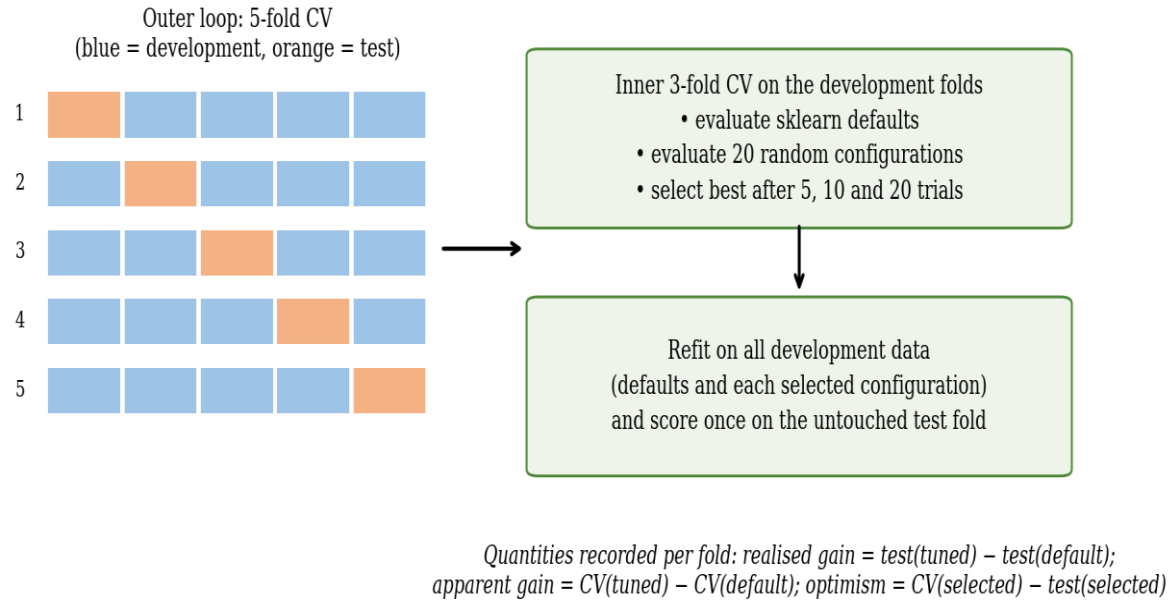


Figure 1. Nested Evaluation Protocol. Hyperparameters are Selected Using Only the Development Folds of Each Outer Split; the Defaults and the Selected Configurations are Then Compared on the Untouched Outer Test Fold

3.4 Quantities of Interest

For each outer fold, three quantities were computed. The realised gain is the balanced accuracy of the tuned configuration on the outer test fold minus that of the default on the same fold. The apparent gain is the best inner cross-validation score found by the search minus the inner score of the default, which is

the improvement a researcher would observe if only the tuning scores were inspected. The optimism of a configuration is its inner score minus its outer test score. The difference between apparent and realised gain equals the optimism of the selected configuration minus the optimism of the default: [1]

Both terms can be non-zero. The selected configuration's inner score is inflated by selection among noisy estimates [12], [13], whereas every inner score, including the default's, is computed with models trained on two-thirds of the development data and therefore tends to underestimate performance on the outer fold.

3.5 Statistical Analysis and Implementation

Realised gains were tested against zero, and apparent gains against realised gains, with the Wilcoxon signed-rank test [35] over the 35 outer folds of each model, with Holm correction over the ten per-model tests [36], and over all 175 folds pooled. Folds of the same dataset share data and the pooled analysis combines different models, so p-values are interpreted together with effect sizes and counts [37], [38]. All experiments were run in Python 3 with scikit-learn 1.8.0 on a single CPU core with fixed random seeds. A full run of defaults and 20 tuning trials with 3-fold inner validation took on average 0.5 s per outer fold for LR, 1.1 s for SVM, 16.1 s for RF, 19.6 s for MLP and 21.4 s for GB.

4. RESULTS AND DISCUSSION

4.1 Realised Gain on Held-Out Data

Table 3 lists the held-out balanced accuracy of the default and tuned configurations for each dataset and model, and Figure 2 shows the realised gain. The gains were small and inconsistent in sign. The largest improvements were for SVM on Sonar (+3.01 percentage points) and Pima (+2.38), GB on Cleveland (+3.34) and LR on Sonar (+2.49); the largest losses were for GB on Seeds (−1.90), RF on Ionosphere (−1.56), SVM on Cleveland (−1.38) and MLP on Pima (−1.04). Of the 35 dataset–model combinations, tuning improved the mean held-out accuracy in 17 and reduced it in 18 (including cases of no change).

Table 3. Held-Out Balanced Accuracy (%) of the Default → Tuned (20 Trials) Configuration, Averaged over Five Outer Folds

Dataset	LR	SVM	RF	GB	MLP
WDBC	97.5 → 97.3	97.4 → 97.1	95.3 → 95.3	96.5 → 96.6	97.5 → 97.0
Pima	72.0 → 71.8	70.2 → 72.6	71.2 → 72.1	71.7 → 72.1	72.9 → 71.9
Sonar	72.9 → 75.4	84.3 → 87.3	83.6 → 83.1	84.3 → 84.7	84.3 → 85.9
Ionosphere	85.0 → 86.2	92.3 → 91.3	92.8 → 91.2	91.9 → 91.9	89.3 → 89.5
Cleveland	83.0 → 82.7	82.0 → 80.7	78.4 → 80.0	78.2 → 81.6	80.3 → 81.7
Seeds	93.8 → 93.8	93.8 → 94.3	91.0 → 91.0	91.4 → 89.5	94.3 → 95.2
Digits	96.9 → 96.9	98.0 → 97.9	97.5 → 97.5	97.3 → 97.7	97.7 → 97.9

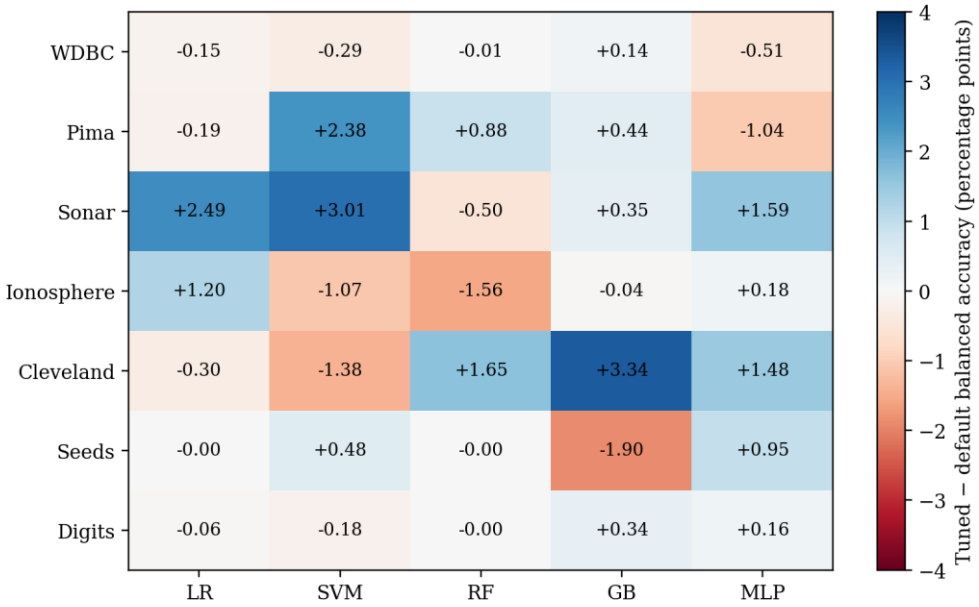


Figure 2. Realised Gain from Tuning: Held-Out Balanced Accuracy of the Tuned Configuration Minus That of the Default, in Percentage Points, Averaged over Five Outer Folds. Blue Cells Indicate Improvement and Red Cells Deterioration

Aggregated by model Table 4, the mean realised gain ranged from 0.07 percentage points for RF to 0.43 for LR, with standard deviations of 2.0–3.7 points across folds, an order of magnitude larger than the means. Tuning improved held-out accuracy in 12–18 of the 35 folds per model, left it unchanged in 2–14 and reduced it in the remainder. None of the per-model gains was significant (Wilcoxon p = 0.455–0.853 before correction). Pooled over all 175 comparisons, the realised gain was 0.34 percentage points; tuning improved accuracy in 76 comparisons, left it unchanged in 30 and reduced it in 69 (p = 0.319).

Table 4. Apparent and Realised Gains from Tuning (20 Random-Search Trials) by Model, in Percentage Points of Balanced Accuracy (35 Outer Folds per Model)

Model	Realised Gain, Mean ± SD	Folds Improved / Unchanged / Worse	P (Realised Gain)	Apparent Gain, Mean ± SD	Runs in Which Inner CV Favoured Tuning	P (Apparent vs Realised), Unadjusted / Holm
LR	0.43 ± 2.45	12 / 14 / 9	0.455	1.21 ± 1.53	83%	0.050 / 0.397
SVM	0.42 ± 3.34	15 / 2 / 18	0.580	1.35 ± 1.82	77%	0.073 / 0.508
RF	0.07 ± 2.01	16 / 5 / 14	0.853	0.66 ± 1.24	66%	0.377 / 1.000
GB	0.38 ± 3.74	18 / 5 / 12	0.742	1.75 ± 1.51	89%	0.010 / 0.096
MLP	0.40 ± 3.46	15 / 4 / 16	0.710	1.71 ± 1.39	100%	0.030 / 0.270
All	0.34	76 / 30 / 69	0.319	1.34	83%	2.2 × 10−5

4.2 Apparent Versus Realised Gain

Judged by the inner cross-validation that drove the search, tuning looked consistently beneficial Figure 3. The best configuration found after 20 trials had a higher inner score than the default in 66% of runs for RF, 77–89% for SVM, LR and GB, and every run for the MLP, and the mean apparent gain was 0.66–1.75 percentage points. On the held-out folds, the corresponding realised gains were 0.07–0.43 points. Pooled over all models, the apparent gain (1.34 points) was about four times the realised gain (0.34 points),

a difference that was highly significant (p = 2.2 × 10−5). Per model, the difference was largest for GB and MLP (unadjusted p = 0.010 and 0.030) but did not remain significant after Holm correction over ten tests. Across the 35 dataset–model combinations, apparent and realised gains were only moderately correlated (Spearman ρ = 0.50), so a large apparent gain was an unreliable indicator of a large real one.

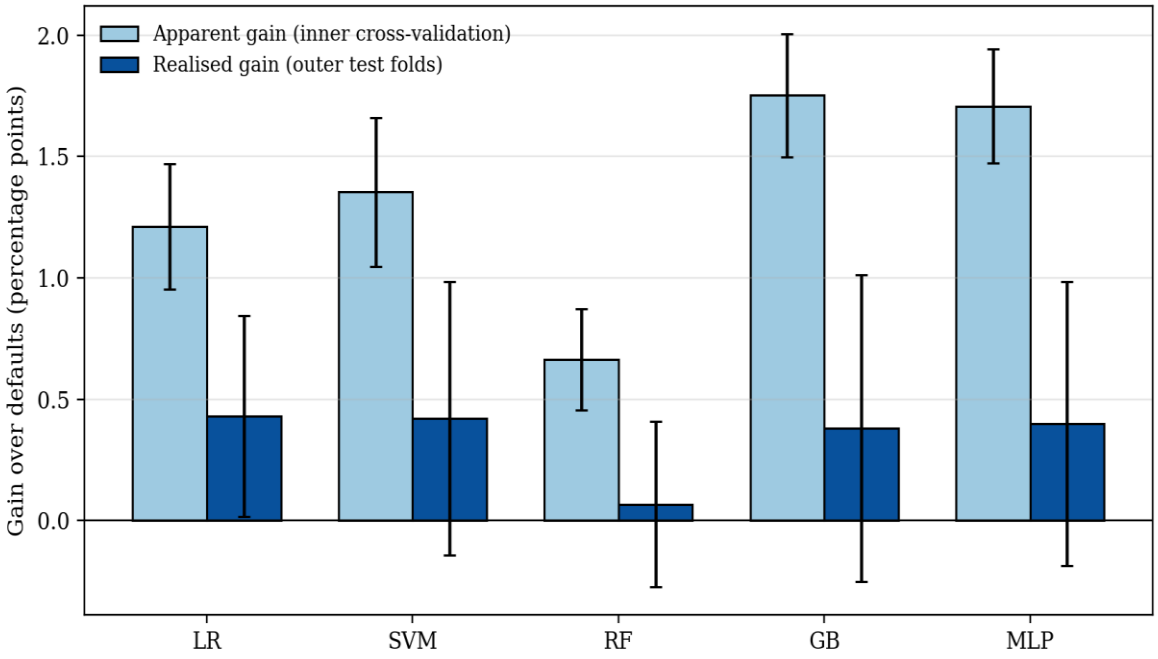


Figure 3. Apparent Gain from Tuning Measured by Inner Cross-Validation (Light Bars) and Realised Gain on the Held-Out Outer Folds (Dark Bars), by Model. Error Bars Show One Standard Error over 35 Outer Folds

Table 5 decomposes the discrepancy. The inner score of the selected configuration overestimated its held-out accuracy only slightly on average (optimism 0.00–0.54 points), whereas the inner score of the default underestimated its held-out accuracy (−0.29 to −1.29 points). The pessimism of the default's inner score is expected, because inner models are trained on two-thirds of the development data. The selected configuration suffers from the same pessimism, but it is offset by the optimism of choosing the maximum of 20 noisy estimates. The net effect, a differential bias of 0.60–1.37 points in favour of the tuned configuration, is the part of the apparent gain that does not exist on new data.

Table 5. Decomposition of the Difference between Apparent and Realised Gain (Percentage Points, Mean over 35 Outer Folds). Optimism = Inner CV Score − Held-Out Score

Model	Optimism of Selected Configuration	Optimism of Default	Differential Bias (Apparent − Realised Gain)
LR	+0.49	−0.29	+0.78
SVM	+0.00	−0.93	+0.93
RF	+0.26	−0.34	+0.60
GB	+0.54	−0.84	+1.37
MLP	+0.02	−1.29	+1.31
All	+0.26	−0.74	+1.00

4.3 Effect of Search Budget

Figure 4 shows how the best inner score evolved with the number of trials. For LR, GB and MLP, random search overtook the default inner score within the first two trials; for SVM and RF, randomly drawn configurations were initially much worse than the defaults, and the mean best inner score exceeded the default only after 7 (SVM) and 9 (RF) trials. The first random SVM configuration was on average 26.1 points

below the default, illustrating that the default kernel width, which scales with the number and variance of the features, is a well-informed choice.

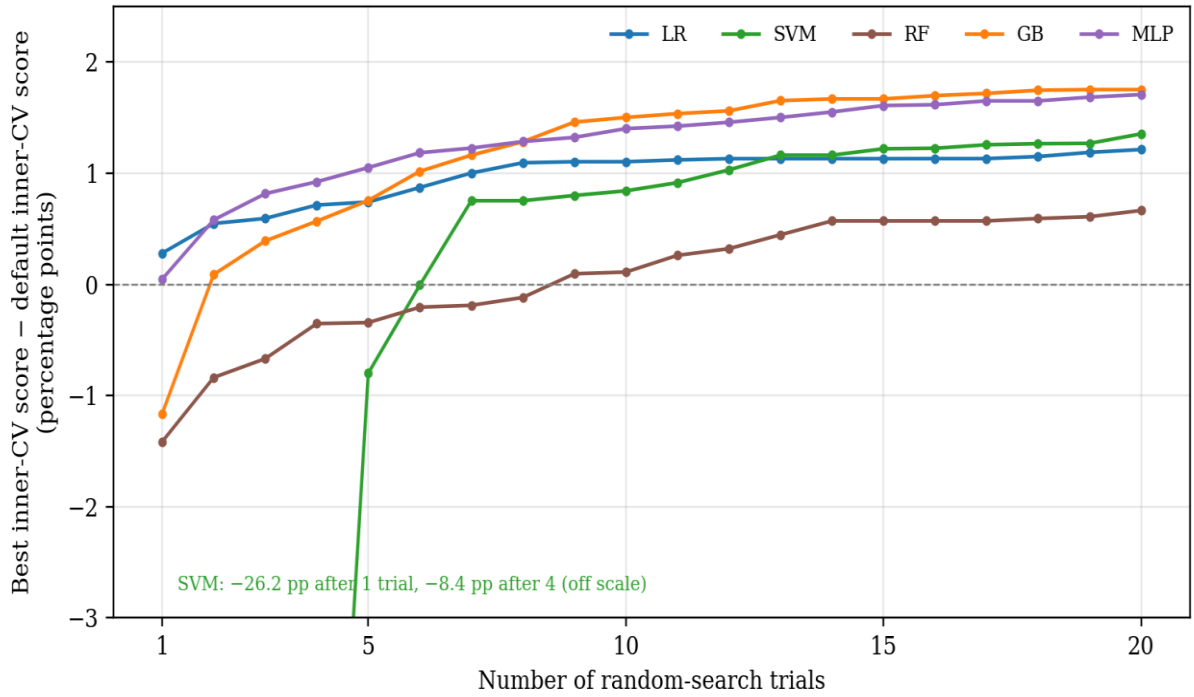


Figure 4. Mean Best Inner Cross-Validation Score Found After Each Number of Random-Search Trials, Relative To the Inner Score of the Default Configuration (35 Runs Per Model). The Y-Axis Is Truncated; the First SVM Trials Lie Below the Plotted Range

On the held-out folds, however, a larger budget did not translate into consistently larger gains Table 6 and Figure 5. For GB, the realised gain was largest after only 5 trials (+0.94 points) and decreased with further search; for MLP it peaked at 10 trials. For SVM and RF, stopping after 5 or 10 trials produced configurations that were worse than the defaults on held-out data, and only the 20-trial searches recovered small positive gains. More search increased the apparent gain steadily, because the maximum of more noisy estimates is higher, while the realised gain fluctuated around a small value.

Table 6. Mean Held-Out Balanced Accuracy (%) over All 35 Outer Folds for the Default and for the Configuration Selected After 5, 10 and 20 Random-Search Trials

Model	Default	5 Trials	10 Trials	20 Trials
LR	85.88	85.73	86.22	86.31
SVM	88.31	87.29	88.21	88.73
RF	87.10	86.35	86.25	87.17
GB	87.34	88.28	87.67	87.72
MLP	88.04	88.59	88.68	88.44

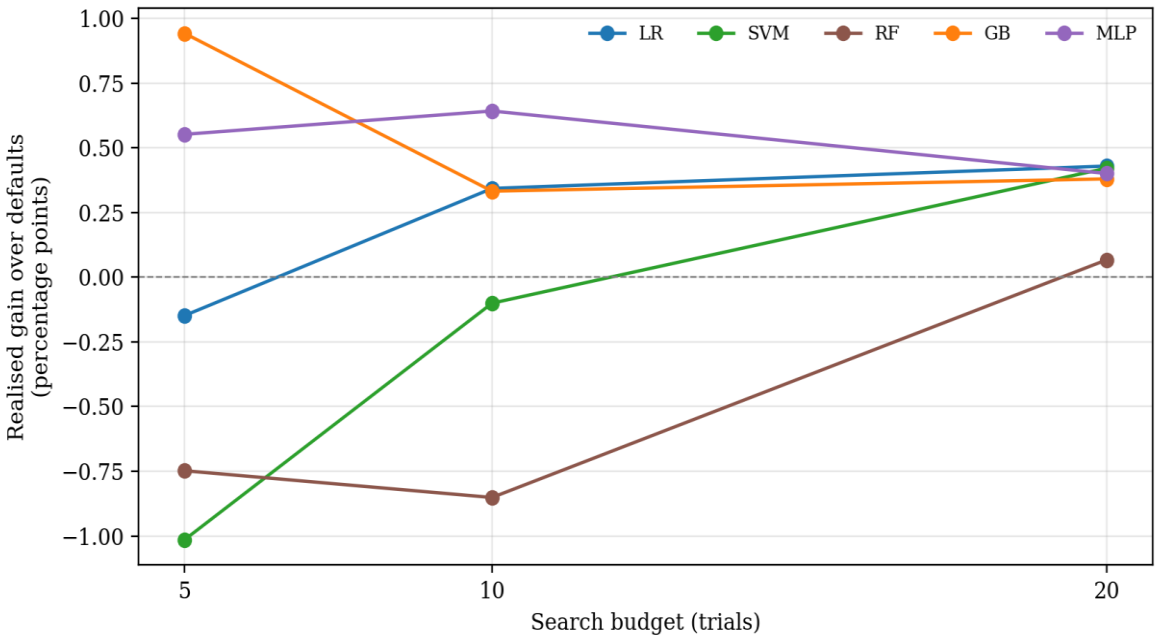


Figure 5. Realised Gain Over The Default on Held-Out Folds as a Function of the Random-Search Budget. Unlike the Apparent Gain, the Realised Gain Does Not Increase Consistently With More Trials

4.4 Does Tuning Change the Best Model?

A practitioner is usually interested in the best model for a dataset rather than in the gain for each algorithm. Table 7 compares the best default model with the best tuned model on each dataset, as judged by mean held-out accuracy. Tuning produced a better best model on two datasets, Sonar (+3.0 points, SVM) and Seeds (+0.9 points, MLP). On the other five datasets, the best tuned model was 0.1–0.9 points worse than the best default model, and on WDBC and Cleveland the best model, LR, was the same with or without tuning. The identity of the best model changed on four datasets, which shows that tuning-based model rankings are themselves unstable on small samples.

Table 7. Best Model per Dataset by Mean Held-Out Balanced Accuracy (%), Without and With Tuning

Dataset	Best Default Model	Accuracy	Best Tuned Model	Accuracy	Difference
WDBC	LR	97.5	LR	97.3	−0.2
Pima	MLP	72.9	SVM	72.6	−0.3
Sonar	MLP	84.3	SVM	87.3	+3.0
Ionosphere	RF	92.8	GB	91.9	−0.9
Cleveland	LR	83.0	LR	82.7	−0.3
Seeds	MLP	94.3	MLP	95.2	+0.9
Digits	SVM	98.0	MLP	97.9	−0.1

4.5 Discussion

4.5.1 Principal Findings

On seven small tabular datasets, random-search tuning of five standard classifiers produced an average held-out improvement of about one-third of a percentage point of balanced accuracy, with a spread across folds roughly ten times larger than the mean and no significant effect for any model. The inner cross-validation that guided the search reported gains about four times larger and favoured the tuned configuration in most runs. These results are consistent with studies finding that defaults of modern libraries are often near-optimal [9], [11] and with the selection-bias literature [12], [13]; the contribution of this study is to measure both effects on the same folds and to show that their combination makes tuning appear substantially more useful than it is when gains are reported from tuning scores.

4.5.2 Why the Apparent Gain Is Inflated

The mechanism is simple. Cross-validation scores on small datasets are noisy, with fold-to-fold standard deviations of several percentage points. When 20 configurations are compared on such scores, the highest score is partly the best configuration and partly the luckiest one. The default, evaluated once, enjoys no such selection. At the same time, all inner scores are pessimistic, because each inner model sees only two-thirds of the development data. The pessimism affects default and tuned configurations alike, but only the tuned configuration receives the compensating selection optimism. The net result is a systematic bias in favour of tuning whenever tuned and default configurations are compared on the tuning scores, which grows with the number of configurations tried. This explains why larger budgets increased the apparent gain but not the realised gain.

4.5.3 Implications for Practice and Reporting

Four recommendations follow. First, improvements from tuning should be reported only from data that played no part in the selection, using nested cross-validation or a separate test set [12], [13]; comparing a tuned configuration's cross-validation score with an untuned default's score overstates the benefit. Second, default configurations should always be reported as a baseline alongside tuned results, with the tuning budget and search space stated [18], [19]. Third, for small tabular datasets, the effort spent on tuning should be weighed against its modest expected benefit; it may be more productive to compare several well-configured default models, or to tune only the few hyperparameters known to matter most [9], [10]. Fourth, when a tuned model is reported to beat a competing method, the variability of the gain across folds should be shown, because on small samples the differences between methods are often smaller than their variability [15], [18].

4.5.4 Limitations

The study considered seven datasets with at most 1,797 samples; on larger datasets cross-validation estimates are less noisy, selection bias is smaller and tuning may yield larger and more reliable gains. Only random search with a budget of 20 configurations and 3-fold inner validation was examined; model-based optimisers [2], [3], larger budgets or repeated inner cross-validation could change the balance between apparent and realised gains. The search spaces were chosen to be reasonable rather than optimal, and hyperparameters not included in them (for example, the number of trees or network depth) were left at fixed values. Five outer folds per dataset give limited power to detect small gains, so the absence of significance should not be read as evidence that tuning never helps; the effect sizes indicate that, where it helps, the benefit is small relative to the noise. Finally, balanced accuracy was the only metric; tuning for probabilistic metrics such as log-loss may behave differently.

5. CONCLUSION

This study measured how much hyperparameter tuning improves five standard classifiers on seven small tabular datasets when the improvement is evaluated on data untouched by the search. Random search over 20 configurations improved held-out balanced accuracy by 0.34 percentage points on average, improved it in fewer than half of 175 comparisons, and changed the best available model for the better on only two of seven datasets. The inner cross-validation used to select the configurations reported a gain of 1.34 points, about four times the realised gain, because the selected configuration benefits from selection optimism that the default does not receive. On small datasets, library defaults are therefore strong baselines, larger tuning budgets mainly increase the apparent rather than the real gain, and any claimed benefit of tuning should be established by nested evaluation. Future work will extend the analysis to larger datasets, model-based optimisers, repeated inner validation and probabilistic performance measures.

Acknowledgments

The authors thank the creators and maintainers of the datasets used in this study, the UCI Machine Learning Repository, and the developers of scikit-learn, as well as the researchers whose work is cited in this article.

Funding Information

This research has received no external funding.

Author Contribution Statement

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Parvinder S. Sandhu	✓	✓	✓		✓				✓	✓	✓	✓		
Jhosep Sánchez-Flores		✓		✓		✓		✓			✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

Conflict of Interest Statement

No conflict of interest.

Informed Consent

Not Applicable.

Ethical Approval

Not Applicable. The study used only publicly available, de-identified datasets.

Data Availability

This study draws on publicly available data from the UCI Machine Learning Repository and the datasets distributed with scikit-learn. No new datasets were created. The analysis code is available from the corresponding author on reasonable request.

REFERENCES

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012. jmlr.org/papers/v13/bergstra12a.html

[2] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in Advances in Neural Information Processing Systems 24, 2011, pp. 2546–2554. papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization

[3] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in Advances in Neural Information Processing Systems 25, 2012, pp. 2951–2959. doi.org/10.48550/arXiv.1206.2944

[4] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," Journal of Machine Learning Research, vol. 18, no. 185, pp. 1–52, 2018. jmlr.org/papers/v18/16-558.html

[5] M. Feurer and F. Hutter, "Hyperparameter optimization," in Automated Machine Learning: Methods, Systems, Challenges, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer, 2019, pp. 3–33. doi.org/10.1007/978-3-030-05318-5_1

[6] B. Bischl et al., "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," WIREs Data Mining and Knowledge Discovery, vol. 13, no. 2, p. e1484, 2023. doi.org/10.1002/widm.1484

[7] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," in Proc. 6th Int. Conf. Learning Representations, 2018. doi.org/10.48550/arXiv.1707.05589

[8] Z. C. Lipton and J. Steinhardt, "Troubling trends in machine learning scholarship," ACM Queue, vol. 17, no. 1, pp. 45–77, 2019. doi.org/10.1145/3317287.3328534

[9] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," Journal of Machine Learning Research, vol. 20, no. 53, pp. 1–32, 2019. jmlr.org/papers/v20/18-444.html

[10] J. N. van Rijn and F. Hutter, "Hyperparameter importance across datasets," in Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2018, pp. 2367–2376. doi.org/10.1145/3219819.3220058

[11] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," arXiv preprint arXiv:2007.07588, 2020. doi.org/10.48550/arXiv.2007.07588

[12] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," BMC Bioinformatics, vol. 7, p. 91, 2006. doi.org/10.1186/1471-2105-7-91

[13] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010. jmlr.org/papers/v11/cawley10a.html

[14] M. Stone, "Cross-validatory choice and assessment of statistical predictions," Journal of the Royal Statistical Society: Series B (Methodological), vol. 36, no. 2, pp. 111–147, 1974. doi.org/10.1111/j.2517-6161.1974.tb00994.x

[15] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," Journal of Machine Learning Research, vol. 15, pp. 3133–3181, 2014. jmlr.org/papers/v15/delgado14a.html

[16] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in Advances in Neural Information Processing Systems 35, Datasets and Benchmarks Track, 2022. doi.org/10.48550/arXiv.2207.08815

[17] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," Information Fusion, vol. 81, pp. 84–90, 2022. doi.org/10.1016/j.inffus.2021.11.011

[18] X. Bouthillier et al., "Accounting for variance in machine learning benchmarks," in Proc. Machine Learning and Systems, vol. 3, 2021, pp. 747–769. doi.org/10.48550/arXiv.2103.03098

[19] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith, "Show your work: Improved reporting of experimental results," in Proc. 2019 Conf. Empirical Methods in Natural Language Processing, 2019, pp. 2185–2194. doi.org/10.18653/v1/D19-1224

[20] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in Proc. SPIE 1905, Biomedical Image Processing and Biomedical Visualization, 1993, pp. 861–870. doi.org/10.1117/12.148698

[21] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in Proc. Annu. Symp. Computer Applications in Medical Care, 1988, pp. 261–265. www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/

[22] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," Neural Networks, vol. 1, no. 1, pp. 75–89, 1988. doi.org/10.1016/0893-6080(88)90023-8

[23] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," Johns Hopkins APL Technical Digest, vol. 10, no. 3, pp. 262–266, 1989. archive.ics.uci.edu/dataset/52/ionosphere

[24] R. Detrano et al., "International application of a new probability algorithm for the diagnosis of coronary artery disease," The American Journal of Cardiology, vol. 64, no. 5, pp. 304–310, 1989. doi.org/10.1016/0002-9149(89)90524-9

[25] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Żak, "Complete gradient clustering algorithm for features analysis of X-ray images," in Information Technologies in Biomedicine, Advances in Intelligent and Soft Computing, vol. 69. Berlin: Springer, 2010, pp. 15–24. doi.org/10.1007/978-3-642-13105-9_2

[26] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits," UCI Machine Learning Repository, 1998. archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits

[27] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. jmlr.org/papers/v12/pedregosa11a.html

[28] D. R. Cox, "The regression analysis of binary sequences," Journal of the Royal Statistical Society: Series B (Methodological), vol. 20, no. 2, pp. 215–242, 1958. doi.org/10.1111/j.2517-6161.1958.tb00292.x

[29] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995. doi.org/10.1007/BF00994018

[30] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001. doi.org/10.1023/A:1010933404324

[31] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," The Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001. doi.org/10.1214/aos/1013203451

[32] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems 30, 2017, pp. 3146–3154. papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, 1986. doi.org/10.1038/323533a0

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. Learning Representations, 2015. doi.org/10.48550/arXiv.1412.6980

[35] F. Wilcoxon, "Individual comparisons by ranking methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, 1945. doi.org/10.2307/3001968

[36] S. Holm, "A simple sequentially rejective multiple test procedure," Scandinavian Journal of Statistics, vol. 6, no. 2, pp. 65–70, 1979. www.jstor.org/stable/4615733

[37] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006. jmlr.org/papers/v7/demsar06a.html

[38] C. Nadeau and Y. Bengio, "Inference for the generalization error," Machine Learning, vol. 52, no. 3, pp. 239–281, 2003. doi.org/10.1023/A:1024068626366

How to Cite: Parvinder S. Sandhu, Jhosep Sánchez-Flores (2026). Tuned on paper, not in practice: How much does hyperparameter tuning improve classifiers on small tabular datasets? Journal of Artificial Intelligence, Machine Learning and Neural Network, 6(2). 120-133 https://doi.org/10.55529/jaimlnn.62.120.133

BIOGRAPHIES OF AUTHORS



Parvinder S. Sandhu, is a Professor and Director-Principal, formerly of the Rayat-Bahra Institute of Engineering and Bio-Technology, Mohali, Punjab, India. He has a long record in computer science and engineering, with research in machine learning, software engineering and quality prediction. His interests include the empirical evaluation of learning algorithms and the design of experimental protocols that make comparisons between models fair and reproducible.



Jhosep Sánchez-Flores, is a researcher in the Escuela Profesional de Ingeniería de Sistemas e Informática, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional de San Martín, Tarapoto, Peru. His interests lie in applied machine learning and systems engineering, with attention to the empirical evaluation of learning algorithms on real data. He works on reproducible experimental comparison of classifiers and on the practical cost of model selection and tuning in resource-constrained settings.