MSBoost: Using Model Selection with Multiple Base Estimators for Gradient Boosting

Agnij Moitra
Amity International School (Saket)
agnijmoitra[at]outlook[dot]com

Abstract

Gradient boosting is a widely used machine learning algorithm for tabular regression, classification and ranking. Although, most of the open source implementations of gradient boosting such as XGBoost, LightGBM and others have used decision trees as the sole base estimator for gradient boosting. This paper, for the first time, takes an alternative path of not just relying on a static base estimator (usually decision tree), and rather trains a list of models in parallel on the residual errors of the previous layer and then selects the model with the least validation error as the base estimator for a particular layer. This paper has achieved *state-of-the-art* results when compared to other gradient boosting implementations on 50+ tabular regression and classification datasets. Furthermore, ablation studies show that MSBoost is particularly effective for small and noisy datasets. Thereby, it has a significant social impact especially in tabular machine learning problems in the domains where it is not feasible to obtain large high quality datasets.

1 Introduction

Gradient boosting [1, 2] has been a powerful boosting [3] based machine learning algorithm that has achieved state-of-the-art accuracy in various real world tasks. Such as in particle physics, biochemistry, finance, fraud detection, search engine recommendations, drug discovery and many others [4–16]. Its significance lies in its ability to handle diverse data types and complex feature engineering whilst effectively managing high-dimensional, noisy datasets with heterogeneous features.

It builds a 'stronger' predictive model by combining several weaker models through an iterative greedy process that focuses on correcting the errors of previous models, which is based on sound theoretical evidence as per [17]. Popular implementations of gradient boosting include XGBoost [18], which enhances traditional methods by introducing regularization to prevent overfitting and tree pruning to improve efficiency, and LightGBM [19], which differs by using a leaf-wise tree growth strategy instead of level-wise growth, and implements Gradient-based One-Side Sampling (GOSS) to speed up training on large datasets while maintaining accuracy. Furthermore, other variants include CatBoost [20] which introduces a novel categorical encoding method to mitigate target leakage, and using Artificial Neural Network, Principal Component Analysis and Random Projections for feature extraction and combine this with gradient boosting as per AugBoost [21].

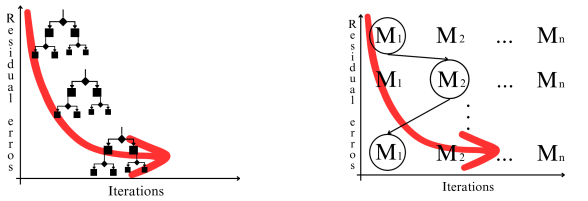
The main contribution of this paper, Model Selection based Gradient Boosting (MSBoost[1]), is to explore, for the first time, the usage of model selection in order to find the base estimator with the least validation error. Unlike the current methods which use a single base estimator, usually decision tree [22–24], although previous research has been done in boosting other models [25]. Benchmarking this method, MSBoost, on 50+ datasets indicate that this method outperforms previous methods such as LightGBM and XGBoost, and based on the ablation studies performed it can be observed

[1] https://github.com/Agnij-Moitra/MSBoost

Preprint. Under review.

that MSBoost is particularly effective for small and noisy datasets. Thereby, MSBoost would be particularly effective for tabular regression and classification problems where it is not feasible or expensive to obtain thousands of high quality samples.

2 Method



(a) Conventional Gradient Boosting

(b) Model Selection based Gradient Boosting (MSBoost)

Figure 1: Conventional Gradient Boosting methods usually use Decision Trees, also known as CART(s), as the sole base estimator in order to minimize the residual errors over a number of iterations. Whereas, MSBoost from a list of ML models dynamically would choose the one with the least residual errors, in parallel, and use it as the base estimator for that layer.

Similar to gradient boosting, the goal of MSBoost is to approximate any arbitrary but particular $\mathcal{F} : \mathbb{R}^m \to \mathbb{R}$ with a series of additive and scaled F_i in order to minimize $\mathcal{L}(\mathcal{F}(\mathbf{x}), F(\mathbf{x}))$. For any given tabular dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and a differentiable loss function $\mathcal{L}(\mathbf{y}, F(\mathbf{x}))$. Wherein $\mathbf{x}_i$ is an arbitrary but particular vector $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^m)$ containing m features, and $\mathbf{y} \in \mathbb{R}^n$, which has n samples is the target vector. First, MSBoost initializes the first estimator as a constant term i.e $F_0(\mathbf{x}) = \arg\min_k \sum_{i=1}^n \mathcal{L}(y_i, k)$, which turns out to be the arithmetic mean of the target values vector $\mathbf{y}$. Next, for each subsequent iteration $i = 1, \ldots, N$ it shall compute the pseudo residuals:

$$\mathbf{r}_i = -\left[\frac{\partial \mathcal{L}(\mathbf{y}, F_{i-1}(\mathbf{x}))}{\partial F_{i-1}(\mathbf{x})}\right] \tag{1}$$

and the base estimator for i^{th} layer is based on a list of models $\mathcal{M}$, such that:

$$h_i(\mathbf{x}) = \arg \min_{\forall M \in \mathcal{M}} \mathcal{L}(\mathbf{y}, M(\mathbf{r}_i))(\mathbf{r}_i) \tag{2}$$

Finally it would update the model for i^{th} layer, i.e $F_i(\mathbf{x}) = F_{i-1}(\mathbf{x}) + \alpha \cdot h_i(\mathbf{x})$, and the final prediction, $\hat{\mathbf{y}} = F(\mathbf{x}) = F_0(\mathbf{x}) + \sum_{i=1}^N F_i(\mathbf{x})$.

2.1 Rationale for Model Selection in Gradient Boosting

Since model selection searches for $\arg\min_{M \in \mathcal{M}} \mathcal{L}(\mathbf{y}, M(\mathbf{r}_i))$ for each iteration i, $\Rightarrow \mathcal{L}(\mathbf{y}, M(\mathbf{r}_i)) \leq \mathcal{L}(\mathbf{y}, S(\mathbf{r}_i)), \forall S$ which are static machine learning models say Decision Tree. And over a large number of iterations N, $\mathbf{r}_{i,\mathcal{M}}$ (model selection, a dynamic method) $<$ $\mathbf{r}_{i,S}$ (for any static base estimator). This is technically a "$\leq$" inequality, but based on the inductive proposition that over a large number of iterations, N, a static method would have higher $\mathbb{E}(\mathbf{r}_i)$ than dynamically selecting base estimators in each iteration, the "$<$" inequality should hold true. Wherein the base case is $\mathbb{E}(\mathbf{r}_{i,\mathcal{M}}) < \mathbb{E}(\mathbf{r}_{i,S})$, which is empirically true as per [26, 27] and theoretically justified by the *No Free Lunch Theorem* [28, 29]. Furthermore, analysing the effect of specific base estimators stacked over N iteration on the residual plots shall be an interesting obsevation, for example a non-linear model like [30] may have a more linear residual plot when compared to that of a linear model, so a non linear base estimator in i^{th} iteration may lead to a linear model in $i + 1^{th}$ iteration. But this has been left for a avenue for future research.

Also, as empirically demonstrated by [26, 27], there is no *one-size-fits-all* baseline model which does well on all types of datasets, which empirically justifies as to why boosting multiple estimators might be effective; and, increase the diversity of the base learners, which potentially help to improve the generalization performance (i.e less variance) [31].

2.2 Model Selection Methods[2]

Naïve Method The naïve way for model selection is to train all the available base estimators on $\mathbf{r}_i$ in parallel. This way would ensure that the model with the least residual errors is truly being selected for each layer and precisely conforms to the theoretical rationale stated in Section 2.1. But this would have the largest time complexity, i.e $O(\text{Number of Iterations} \times \text{Base model with the highest time complexity i.e the limiting factor})$.

Random Sampling Sampling a subset of models from $\mathcal{M}$, shall reduce the overall training time, but this may not find the model with least possible validation residual errors.

Frequency & Probability Based Sampling Assuming that only a subset of models from $\mathcal{M}$ would be used for most of the time due to the characteristics of the dataset being used. For the first I iterations, this shall be a track of the frequency of the top N models, and for the rest of the iterations only train the top N models initially found. Here I and N are hyper-parameters. A more vigorous method for this would be to use Bayesian model selection [32–34] using Dirchlet's distribution [35, 36] and train the models with the top N probabilities of being used. The pseudo-code can be found in Appendix A.

3 Experiments & Discussion

Comparison with baselines MSBoost (random sampling half of the models from $\mathcal{M}$ for training in each iteration) was compared[3] with XGBoost and LightGBM. The source code of the experiments are available, and can be reproduced (`https://github.com/Agnij-Moitra/MSBoost`). Unfortunately due to constrained computational resources the benchmarking was done on 1K samples on OpenML [37, 38] datasets with 0.01 lasso threshold to screen for irrelevant features which would have increased the computational costs. Table 1 compares the mean squared error with 5 fold cross validation (CV), and Table 2 compares the log loss with 5 Fold CV; please check Appendix **??** for entire results. Paired single tailed t-test reveal that MSBoost yields a statistically significant improvement over LightGBM and XGBoost in metrics, with p-value $<< 0.001$ (excluding outliers like wave_energy), and $p < 0.02$ for standard deviation thus improving the bias-variance trade-off [39]. It should be noted even without regularization, and GOSS and EFB of XGBoost and LightGBM respectively, MSBoost has a statistically significant improvement. Thereby, this may have even better improvement over previous methods if those techniques are incorporated in MSBoost.

Table 1: Comparison (regression) with baselines based on mean squared error (MSE)

	MSBoost	**LightGBM**	**XGBoost**
wave_energy	0.0 ± 0.0	1.9e+9 ± 2.9e+8	3.0e+9 ± 4.5e+8
Friedman 2	150 ± 31	385 ± 57	501 ± 58
Sparse Uncorr.	1.0 ± 0.15	1.5 ± 0.11	1.7 ± 0.22
kin8nm	2.1e-2 ± 1e-4	3.1e-2 ± 1.5e-3	3.6e-2 ± 1.3e-3
sarcos	32 ± 8	46 ± 15	48 ± 10
Moneyball	431 ± 24	588 ± 42	635 ± 39
yprop_4_1	7e-4 ± 1e-4	9e-4 ± 1e-4	1.1e-3 ± 1e-4
fps_benchmark	2354 ± 110	2917 ± 104	3758 ± 395
Zurich Transport	10 ± 0.7	12 ± 0.9	15 ± 1.4
Diabetes	3017 ± 333	3590 ± 433	3991 ± 651

Table 2: Comparison (classification) with baselines based on log loss

	MSBoost	**LightGBM**	**XGBoost**
phoneme	0.34 ± 0.03	0.43 ± 0.07	0.43 ± 0.06
guillermo	0.56 ± 0.04	0.69 ± 0.10	0.77 ± 0.11
MagicTelescope	0.40 ± 0.04	0.48 ± 0.05	0.50 ± 0.05
heloc	0.58 ± 0.01	0.67 ± 0.08	0.78 ± 0.09
Bioresponse	0.50 ± 0.02	0.57 ± 0.08	0.59 ± 0.07
electricity	0.54 ± 0.06	0.61 ± 0.08	0.65 ± 0.09
Australian	0.50 ± 0.03	0.54 ± 0.06	0.64 ± 0.08
house_16H	0.38 ± 0.03	0.40 ± 0.07	0.42 ± 0.06
pol	0.17 ± 0.04	0.18 ± 0.05	0.15 ± 0.04
california	0.37 ± 0.03	0.39 ± 0.05	0.40 ± 0.06

Impact of dataset dependent factors Figure 2 highlights how MSBoost and the baseline models perform when noise, number of samples and others are progressively increased on Scikit-Learn's [40] make classification dataset [41]. This is a *cherry-picked* example, but similar trend was found on all other Scikit-Learn's synthetic datasets, their plots can be found in Appendix B.2. Using paired single tail t-test that MSBoost has a p-value < 0.01 when compared to XGBoost and LightGBM for robustness against noise and for impact of number of samples when compared to the baseline models.

[2]The model selection was done on a validation dataset, subsampled from the training data.

[3]For now it wasn't compared to CatBoost, since in order to have a fair comparison, since MSBoost's implementation doesn't have targeted feature encoding for now.

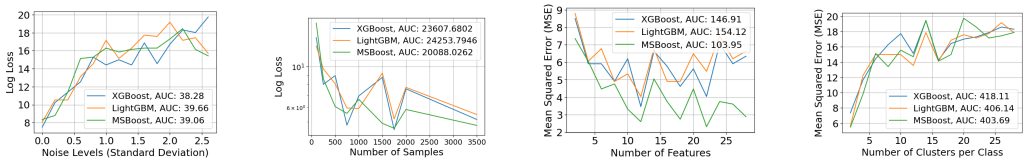


Figure 2: Impact of dataset dependent various factors on log loss for Make Classification Dataset [41]

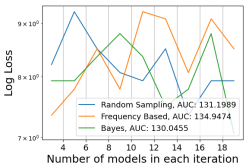


Figure 3: Impact of changing number of models trained, for model selection methods (Scikit-Learn's Make Classification dataset)

Table 3: Comparison on small noisy real world datasets with significant social impact (†MSE & ‡Log Loss)

	UCI ID [42]	**MSBoost**	**LightGBM**	**XGBoost**
AIDS Clinical Trials†	890	8.1e-2 ± 6.5e-3	+7.1% ± +33.8%	+13.5% ± -1.5%
Student Performance†	320	6.3 ± 0.81	+5.4% ± +15.3%	+25.9% ± +42.2%
Energy Efficiency†	242	1.7 ± 0.18	+2.2% ± +16.9%	+34.3% ± +19.1%
Diabetes†	[40]	3017 ± 333	+18.9% ± +30.0%	+32.9% ± +95.4%
Liver Disorders†	60	9.6 ± 1.19	+4.7% ± +2.6%	+23.1% ± -0.3%
Heart Failure Clinical Records†	519	0.12 ± 0.03	+3.4% ± +35.95%	+17.9% ± +37.1%
Thyroid Cancer Recurrence‡	915	1.2 ± 0.88	+30.6% ± -13.1%	+22.9% ± +4.1%
Rice (Cammeo and Osmancik)‡	545	2.79 ± 0.10	+7.7% ± +130.3%	+6.1% ± +31.0%
Blood Transfusion Service‡	176	8.3 ± 0.48	+10.4% ± +0.3%	+13.9% ± +148.0%
Acute Inflammations‡	184	0.0 ± 0.0	0.3 ± 0.6	0.75 ± 1.2
SPECTF Heart‡	96	6.4 ± 1.24	+4.2% ± +93.0%	+2.1% ± +34.9%
Glioma Grading Clinical & ...‡	759	4.8 ± 0.75	+33.9% ± +20.5%	+34.8% ± -26.7%

Impact of model selection methods The effect of number of base models trained on the model selection methods is demonstrated in Figure 3, this is a *cherry-picked* example the rest can be found in Appendix B.3. There is no statistically significant difference in choosing the bayes method over the frequency based method (p = 0.28), but the bayes method turns out to be better than random sampling (p = 0.06).

Social Impact As mentioned above, there is statistically significant evidence that using model selection along with gradient boosting, MSBoost, may improve bias-variance trade-off. Particularly on small and noisy datasets, where usually other machine learning algorithms tend to overfit [43, 44]. Table 3 demonstrates a few possible tabular regression and classification problems with significant social impact, where MSBoost turns out to be better than other methods in terms of MSE/log loss and standard deviation (5 Fold CV).

Limitations (i) Since it trains multiple models for each iteration, MSBoost, has a enormously high time complexity. Where the limiting factor is SVM's RBF kernel, which is quadratic. So the worst case time complexity of MSBoost is approximately $O(n^2)$, whereas LightGBM and others have a time complexity of $O(n\ log\ n)$ (ii) Due to system resource constrains (AMD Ryzen 5 3550H & 8 GB RAM, Ubuntu 20.4 & 22.04.4 LTS), and the enormous time complexity the test most of the benchmarking couldn't be done for more than 1K samples, although this was compensated by benchmarking on 50+ datasets with 5 fold CV.

4 Conclusion

This paper introduces a novel gradient boosting method, MSBoost, which uses model selection to find base estimators for each iteration of gradient boosting. Empirical results show that there is a statistically significant evidence that this method outperforms other popular gradient boosting methods (LightGBM & XGBoost), both in terms of errors and standard deviation of the error. Furthermore, ablation studies reveal that MSBoost outperforms other methods on (synthetic & real) small and noisy datasets, a domain where machine learning algorithms usually struggle. Future work, shall incorporate techiques like targeted feature encoding, GOSS, EFB and other from the current Gradient Boosting methods.

Acknowledgments and Disclosure of Funding

In no particular order, Dr. Arpan Mukherjee (Research Scientist, University at Buffalo) is gratefully acknowledged for his suggestions and reviewing this research. Deepika Pareek (Head of Department, Computer Science at Birla Vidya Niketan) and Rudransh Agnihotri (Founder and CEO of FuturixAI and Quantum Works) are thanked with gratitude for their guidance and encouragement for an initial starting point of this research, which was presented at Regeneron International Science and Engineering Fair (2023). Also parts of the math notation and the table style was adapted from CatBoost [20]. This is a bootstrapped research, and didn't receive any external funding, and the author has no competing financial interests.

References

1. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of Statistics,* 1189–1232 (2001).
2. Friedman, J. H. Stochastic gradient boosting. *Computational Statistics & Data Analysis* **38,** 367–378 (2002).
3. Schapire, R. E. The strength of weak learnability. *Machine Learning* **5,** 197–227 (1990).
4. Chen, T. & He, T. *Higgs boson discovery with boosted trees* in *NIPS 2014 workshop on high-energy physics and machine learning* (2015), 69–80.
5. Wu, Z. *et al.* MoleculeNet: a benchmark for molecular machine learning. *Chemical science* **9,** 513–530 (2018).
6. Nobre, J. & Neves, R. F. Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. *Expert Systems with Applications* **125,** 181–194 (2019).
7. Hajek, P., Abedin, M. Z. & Sivarajah, U. Fraud detection in mobile payment systems using an XGBoost-based framework. *Information Systems Frontiers* **25,** 1985–2003 (2023).
8. Burges, C. J. From ranknet to lambdarank to lambdamart: An overview. *Learning* **11,** 81 (2010).
9. Li, P., Wu, Q. & Burges, C. *McRank: Learning to Rank Using Multiple Classification and Gradient Boosting* in *Advances in Neural Information Processing Systems* (eds Platt, J., Koller, D., Singer, Y. & Roweis, S.) **20** (Curran Associates, Inc., 2007). `https://proceedings.neurips.cc/paper_files/paper/2007/file/b86e8d03fe992d1b0e19656875ee557c-Paper.pdf`.
10. Gulin, A., Kuralenok, I. & Pavlov, D. *Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank* in *Proceedings of the Learning to Rank Challenge* (eds Chapelle, O., Chang, Y. & Liu, T.-Y.) **14** (PMLR, Haifa, Israel, July 2011), 63–76. `https://proceedings.mlr.press/v14/gulin11a.html`.
11. Sikander, R., Ghulam, A. & Ali, F. XGB-DrugPred: computational prediction of druggable proteins using eXtreme gradient boosting and optimized features set. *Scientific reports* **12,** 5505 (2022).
12. Sun, X., Liu, M. & Sima, Z. A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters* **32,** 101084. ISSN: 1544-6123. `https://doi.org/10.1016/j.frl.2018.12.032` (2020).
13. Natekin, A. & Knoll, A. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics* **7,** 21 (2013).
14. Roe, B. P. *et al.* Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **543,** 577–584 (2005).
15. Wu, Q., Burges, C. J., Svore, K. M. & Gao, J. Adapting boosting for information retrieval measures. *Information Retrieval* **13,** 254–270 (2010).
16. Zhang, Y. & Haghani, A. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies* **58,** 308–324 (2015).
17. Kearns, M. & Valiant, L. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)* **41,** 67–95 (1994).
18. Chen, T. & Guestrin, C. *Xgboost: A scalable tree boosting system* in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), 785–794.

19. Ke, G. *et al.* Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* **30** (2017).
20. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. & Gulin, A. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems* **31** (2018).
21. Tannor, P. & Rokach, L. *AugBoost: Gradient Boosting Enhanced with Step-Wise Feature Augmentation* in *Twenty-Eighth International Joint Conference on Artificial Intelligence* (July 2019), 3555–3561. https://doi.org/10.24963/ijcai.2019/493.
22. Li, B., Friedman, J., Olshen, R. & Stone, C. Classification and regression trees (CART). *Biometrics* **40,** 358–361 (1984).
23. Friedman, J., Hastie, T. & Tibshirani, R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics* **28,** 337–407 (2000).
24. Rokach, L. & Maimon, O. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **35,** 476–487 (2005).
25. Zięba, M., Tomczak, J. M., Lubicz, M. & Świątek, J. Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. *Applied soft computing* **14,** 99–108 (2014).
26. Caruana, R. & Niculescu-Mizil, A. *An empirical comparison of supervised learning algorithms* in *Proceedings of the 23rd international conference on Machine learning* (2006), 161–168.
27. McElfresh, D. *et al.* When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems* **36** (2024).
28. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* **1,** 67–82 (1997).
29. Wolpert, D. H. The supervised learning no-free-lunch theorems. *Soft computing and industry: Recent applications,* 25–42 (2002).
30. Cortes, C. & Vapnik, V. Support-vector networks. *Machine learning* **20,** 273–297 (1995).
31. Zhou, Z.-H. *Ensemble methods: foundations and algorithms* (CRC press, 2012).
32. Wasserman, L. Bayesian model selection and model averaging. *Journal of mathematical psychology* **44,** 92–107 (2000).
33. Ando, T. Predictive Bayesian Model Selection. *American Journal of Mathematical and Management Sciences* **31,** 13–38. https://doi.org/10.1080/01966324.2011.10737798 (2011).
34. Chipman, H. *et al.* The practical implementation of Bayesian model selection. *Lecture Notes-Monograph Series,* 65–134 (2001).
35. Dirichlet, P. G. L. Ueber die Darstellung ganz willkürlicher Funktionen durch Sinus-und Cosinusreihen. *Repertorium der Physik, Bd. I, S,* 252–174 (1889).
36. Dirichlet, P. G. L. & Seidel, P. L. *Die darstellung ganz willkürlicher functionen durch sinus-und cosinusreihen* **116** (1900).
37. Vanschoren, J., Van Rijn, J. N., Bischl, B. & Torgo, L. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter* **15,** 49–60 (2014).
38. Bischl, B. *et al.* OpenML: A benchmarking layer on top of OpenML to quickly create, download, and share systematic benchmarks. *NeurIPS.* https://openreview.net/forum?id=OCrD8ycKjG (2021).
39. Briscoe, E. & Feldman, J. Conceptual complexity and the bias/variance tradeoff. *Cognition* **118,** 2–16 (2011).
40. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
41. Guyon, I. *Design of experiments of the NIPS 2003 variable selection benchmark* in *NIPS 2003 workshop on feature extraction and feature selection* **253** (2003), 40.
42. Kelly, M., Longjohn, R. & Nottingham, K. *The UCI Machine Learning Repository* https://archive.ics.uci.edu.
43. Lever, J., Krzywinski, M. & Altman, N. Points of significance: model selection and overfitting. *Nature methods* **13,** 703–705. https://doi.org/10.1038/nmeth.3968 (2016).

44. Oates, T. & Jensen, D. *The Effects of Training Set Size on Decision Tree Complexity* in *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics* (eds Madigan, D. & Smyth, P.) **R1.** Reissued by PMLR on 30 March 2021. (PMLR, Apr. 1997), 379–390. https://proceedings.mlr.press/r1/oates97b.html.