Overtuning in Hyperparameter Optimization

Anonymous[1]

[1]Anonymous Institution

Abstract Hyperparameter optimization (HPO) aims to identify an optimal hyperparameter configuration (HPC) such that the resulting model generalizes well to unseen data. Since directly optimizing the expected generalization error is impossible, resampling techniques like holdout validation or cross-validation are used as proxy measures in HPO. However, this implicitly assumes that the HPC minimizing validation error will also yield the best true generalization performance. Given that our inner validation error estimate is inherently stochastic and depends on the resampling, we study: Can excessive optimization of the validation error lead to a similarly detrimental effect as excessive optimization of the empirical risk of an ML model? This phenomenon, which we refer to as overtuning, represents a form of overfitting at the HPO level. Despite its potential impact, overtuning has received limited attention in the HPO and automated machine learning (AutoML) literature. We first formally define overtuning and distinguish it from related concepts such as meta-overfitting. We then reanalyze large-scale HPO benchmark data, assessing how frequently overtuning occurs and its practical relevance. Our findings suggest that overtuning is more common than expected, although often mild. However, in 10% of cases, severe overtuning results in selecting an HPC whose generalization performance is worse than the default HPC. We further examine how factors such as the chosen performance metric, resampling method, dataset size, learning algorithm, and optimization strategy influence overtuning and discuss potential mitigation strategies. Our results highlight the need to raise awareness of overtuning, particularly in the small-data regime, indicating that further mitigation strategies should be studied.

1 Introduction

Hyperparameter optimization (HPO) is a fundamental concept in modern machine learning (ML) to efficiently optimize the predictive performance of ML models and complex pipelines (Feurer and Hutter, 2019; Bischl et al., 2023), the latter being popular to create full AutoML systems. While resampling-based estimates, such as validation-set-holdout or cross-validation (CV), are commonly used to construct the objective function in HPO, their stochastic nature can lead to surprising effects on unseen test data (Figure 1). In particular, aggressive optimization of noisy validation scores may result in choosing a hyperparameter configuration (HPC) that performs worse on unseen data (Ng, 1997; Cawley and Talbot, 2010; Makarova et al., 2021) – a phenomenon we refer to as *overtuning*. Despite its potential adverse consequences, and although some authors have touched upon this topic in the last 25 years, overtuning has received limited attention in the HPO and AutoML literature and is somewhat underexplored. This paper aims to fill this gap by formally defining overtuning and empirically investigating its prevalence and impact.

Our contributions are as follows: 1) We provide a formal definition of overtuning in HPO, distinguishing it from related concepts such as meta-overfitting and test regret. 2) We reanalyze large-scale HPO benchmark data to quantify how frequently overtuning occurs and assess its practical significance. 3) Through mixed model analyses, we examine how overtuning is influenced by the choice of performance metric, resampling method, dataset size, learning algorithm, and optimization strategy. 4) Finally, we propose and discuss potential mitigation strategies to reduce the risk of overtuning and its extent.

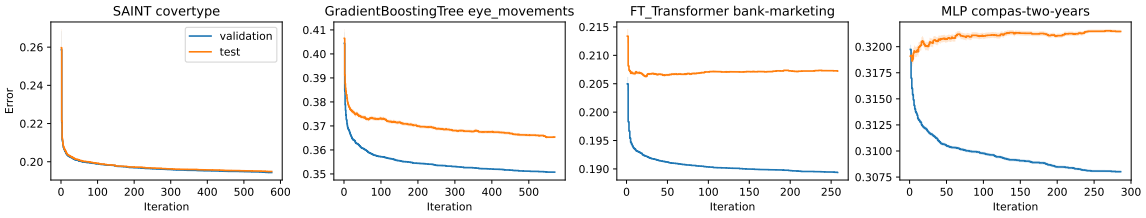


Figure 1: HPO curves from Grinsztajn et al. (2022). Validation performance of incumbents in blue, test performance in orange. From left to right: Ideal, meta-overfitting, benign overtuning, severe overtuning. Ribbons represent standard errors.

2 Problem Statement

Background and notation follows Bischl et al. (2023). The goal of supervised ML is to fit a model given n observations, each sampled from a data generating process $\mathbb{P}_{xy}$, so that it generalizes well to new observations from the same data generating process. An ML learning algorithm or inducer $\mathcal{I}$ configured by a hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ maps a data set $\mathcal{D}$ to a model $\hat{f}$

$$\mathcal{I} : \mathbb{D} \times \boldsymbol{\Lambda} \to \mathcal{H}, \quad (\mathcal{D}, \boldsymbol{\lambda}) \quad \mapsto \hat{f},$$

where $\mathbb{D} := \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$ is the set of all data sets. In the following, we are concerned with the generalization error (GE) of an inducer $\mathcal{I}$ configured by a HPC $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ defined as

$$\mathbb{E}_{\mathcal{D}_{\text{train}} \sim \mathbb{P}_{xy}^n, (\mathbf{x},y) \sim \mathbb{P}_{xy}} \left[L(y, \mathcal{I}_{\boldsymbol{\lambda}}(\mathcal{D}_{\text{train}})(\mathbf{x})) \right], \tag{1}$$

given a training set $\mathcal{D}_{\text{train}}$ of size n_{train} and a loss function L with expectation over data set $\mathcal{D}_{\text{train}}$ sampled from $\mathbb{P}_{xy}^n$ and test sample $(\mathbf{x}, y)$ sampled from $\mathbb{P}_{xy}$. We estimate the GE via $\widehat{\text{GE}}(\mathcal{I}, \boldsymbol{\lambda}, \mathcal{J}, L)$ based on a resampling $\mathcal{J} = ((J_{\text{train},1}, J_{\text{val},1}), \ldots, (J_{\text{train},B}, J_{\text{val},B}))$ with B splits, which leads to the general HPO problem:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \widehat{\text{GE}}(\mathcal{I}, \boldsymbol{\lambda}, \mathcal{J}, L). \tag{2}$$

Here $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_1 \times \ldots \times \boldsymbol{\Lambda}_l$ is the search space containing all hyperparameters for optimization and their ranges where $\boldsymbol{\Lambda}_i$ is a bounded subset of the domain of the ith hyperparameter. The search space may include numeric, integer, and categorical hyperparameters. Hierarchical search spaces can arise when the validity of certain hyperparameters depends on the values of others.

An optimizer sequentially evaluates the ordered sequence of HPCs $(\boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_T)$ with total budget T – we call such a sequence a "trajectory".[1] The ordered incumbent sequence is $(\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_T^*)$. Here, each $\boldsymbol{\lambda}_t^*$ is the validation optimal HPC, if we restrict the selection to the trajectory $(\boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_t)$ up to time point t:

$$\boldsymbol{\lambda}_t^* := \arg\min_{\boldsymbol{\lambda} \in \{\boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_t\}} \widehat{\text{GE}}(\mathcal{I}, \boldsymbol{\lambda}, \mathcal{J}, L).$$

We denote the validation error of an incumbent $\boldsymbol{\lambda}_t^*$ as $\widehat{\text{val}}(\boldsymbol{\lambda}_t^*) := \widehat{\text{GE}}(\mathcal{I}, \boldsymbol{\lambda}_t^*, \mathcal{J}, L)$. We can further denote the true GE of such an optimal $\boldsymbol{\lambda}_t^*$ (fixing the concrete data set $\mathcal{D}_{\text{train}}$ at hand) as:

$$\text{test}(\boldsymbol{\lambda}_t^*) := \text{GE}(\mathcal{I}, \boldsymbol{\lambda}_t^*, \mathcal{D}_{\text{train}}, L) := \mathbb{E}_{(\mathbf{x},y) \sim \mathbb{P}_{xy}} \left[L(y, \mathcal{I}_{\boldsymbol{\lambda}_t^*}(\mathcal{D}_{\text{train}})(\mathbf{x})) \right].$$

We can estimate true GE unbiasedly via another holdout test set or in a nested resampling manner: $\widehat{\text{test}}(\boldsymbol{\lambda}_t^*) := \widehat{\text{GE}}_{\text{unbiased}}(\mathcal{I}, \boldsymbol{\lambda}_t^*, \mathcal{J}_{\text{test}}, L)$.

In this paper we are concerned how we can quantify the effect that overoptimizing on the validation error may decrease true generalization performance of the incumbent, which we will refer to as the overtuning effect.

[1] We use brackets $(\ldots)$ to denote an ordered sequence, whereas $\{\ldots\}$ denotes an unordered set.

3 Characterizing the *Overtuning* Effect

Given a sequence of incumbents, $(\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_t^*)$, we are interested in whether there exists a previous incumbent $\boldsymbol{\lambda}_{t'}^* \in \{\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_t^*\}$, for which $\text{test}(\boldsymbol{\lambda}_{t'}^*) < \text{test}(\boldsymbol{\lambda}_t^*)$ and, by construction, $\widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*) \geq \widehat{\text{val}}(\boldsymbol{\lambda}_t^*)$. In other words, have we already observed an incumbent $\boldsymbol{\lambda}_{t'}^*$ that has lower true GE than the actual incumbent $\boldsymbol{\lambda}_t^*$ at time point t? And would stopping the HPO process early or choosing the incumbent differently have resulted in lower GE? Based on these questions, we introduce the following definition of overtuning and contrast it with meta-overfitting and test regret.

Definition 3.1. Given a trajectory $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T)$, we define for each time point $1 \leq t \leq T$:

$$\textit{overtuning:} \quad \text{ot}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots, \boldsymbol{\lambda}_T) = \text{test}(\boldsymbol{\lambda}_t^*) - \min_{\boldsymbol{\lambda}_{t'}^* \in \{\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_t^*\}} \text{test}(\boldsymbol{\lambda}_{t'}^*) \tag{3}$$

$$\textit{meta-overfitting:} \quad \text{of}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots, \boldsymbol{\lambda}_T) = \text{test}(\boldsymbol{\lambda}_t^*) - \widehat{\text{val}}(\boldsymbol{\lambda}_t^*) \tag{4}$$

$$\textit{trajectory test regret:} \quad \text{tr}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots, \boldsymbol{\lambda}_T) = \text{test}(\boldsymbol{\lambda}_t^*) - \text{test}(\boldsymbol{\lambda}_t^\dagger) \tag{5}$$

$$\textit{oracle test regret:} \quad \text{tr}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots, \boldsymbol{\lambda}_T) = \text{test}(\boldsymbol{\lambda}_t^*) - \text{test}(\boldsymbol{\lambda}_t^{\dagger\dagger}) \tag{6}$$

$$\text{where } \boldsymbol{\lambda}_t^\dagger \coloneqq \operatorname*{arg\,min}_{\boldsymbol{\lambda} \in \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t\}} \text{test}(\boldsymbol{\lambda}) \text{ and } \boldsymbol{\lambda}_t^{\dagger\dagger} \coloneqq \operatorname*{arg\,min}_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \text{test}(\boldsymbol{\lambda}).$$

Overtuning measures the maximum increase in test error between the current incumbent at time point t and any earlier incumbent. In contrast, trajectory test regret compares the current incumbent to all HPCs seen during the search, not just past incumbents. Oracle test regret, on the other hand, quantifies the gap between the current incumbent and the best possible HPC in the entire search space. Since oracle test regret is generally impractical to compute, we refer to trajectory test regret simply as test regret throughout. Lastly, meta-overfitting captures the discrepancy between the observed validation error and the true GE. It directly follows that non-zero meta-overfitting is necessary but not sufficient to observe overtuning (see Appendix B). This is also visualized in Figure 1. While meta-overfitting may seem interesting, it is not central to HPO for several reasons: 1) Validation-test gaps are expected due to finite data and resampling variability. 2) Validation error is mainly used to rank HPCs – its absolute value does not matter. 3) The selected HPC's validation error is a biased estimate of generalization performance anyways. 4) The real concern in HPO is whether we have selected a seemingly strong HPC that underperforms in true generalization, missing out on a previous better alternative.

To make the overtuning measure commensurable across different HPO runs, potentially involving different tasks, performance metrics, and learning algorithms, as well as easier interpretable, it is useful to scale it by the difference of the default test error (this can for example be the test error of $\boldsymbol{\lambda}_1^* = \boldsymbol{\lambda}_1$ or an explicit default HPC) and the best test error observed over incumbents.

Definition 3.2. Given a sequence of HPC evaluations $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T)$, the relative *overtuning* effect at time point t is defined as

$$\tilde{\text{ot}}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T) = \frac{\text{ot}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots, \boldsymbol{\lambda}_T)}{\text{test}(\boldsymbol{\lambda}_1^*) - \min_{\boldsymbol{\lambda}_{t'}^* \in \{\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_t^*\}} \text{test}(\boldsymbol{\lambda}_{t'}^*)} \tag{7}$$

This relative overtuning indicates how much worse the current test error is compared to the maximum possible improvement in true generalization performance achieved by HPO. For example, a value of 0 implies no overtuning, while value of 0.1 indicates a 10% loss in test performance made during HPO due to overtuning. In other words, without overtuning, our test performance could have been 10% better compared to this reference of potential test performance improvement. Values of 1 and above imply that overtuning has resulted in no improvement over the initial test error, and we lost all HPO progress and HPO even degraded generalization performance.

4 Related Work

We discuss related work concerned with notions of overtuning in HPO. An extended discussion is available in Appendix C. Mitigation strategies are discussed in Section 7. Cawley and Talbot (2010) explore overfitting in model selection, highlighting that criteria like CV estimates of GE have a bias and variance due to finite data. High-variance selection criteria can lead to models that excel on validation data but fail to generalize – an observation consistent with our definition of overtuning, although Cawley and Talbot (2010) do not formally define or quantify it. Their synthetic experiments show that validation performance can improve while test performance deteriorates. In real-world settings evidence is limited and they note that a more flexible kernel in kernel ridge regression may overfit validation data compared to a simpler alternative.

Ng (1997) critiques the common practice of selecting models based solely on validation error, noting that the model with the lowest validation error may not have the lowest true GE. This mismatch arises from the variance in the validation error estimator and the sensitivity of the true GE's conditional posterior distribution to the observed validation error conditioned on. This aligns with our definition of overtuning, where validation error may improve while true GE worsens. To address this, Ng (1997) proposes LOOCVCV, which estimates the GE of the best-of-n models for varying n to determine how many models can be considered before overfitting to validation data occurs. The final model is then chosen based on a validation performance percentile k derived from the optimal n. On noisy synthetic data, LOOCVCV outperforms naïve selection, but it can be overly conservative in lower-noise settings.

Makarova et al. (2022) propose an early stopping criterion for Bayesian Optimization (BO) in HPO. We refer to Garnett (2023) for a general introduction to BO and to Feurer and Hutter (2019); Bischl et al. (2023) for an introduction in the context of HPO. Their method combines a confidence bound on the surrogate model's regret and the variance of the CV estimator. This approach reduces computational costs with small impact on generalization performance. They also touch on what we define as overtuning, noting that gains in validation performance might not translate to test improvements due to weak validation-test correlations. A prior workshop version (Makarova et al., 2021) highlighted this more explicitly, observing test performance drops in Elastic Net models trained via SGD despite ongoing validation gains.

Lévesque (2018) addresses what we define as overtuning in HPO, showing empirically that validation performance can improve while test performance deteriorates. In a large-scale support vector machine HPO study on 118 datasets using classification error as performance metric, they explore potential mitigation strategies: reshuffling resampling splits, selecting the incumbent on an outer test set (Dos Santos et al., 2009; Koch et al., 2010; Igel, 2012), and selecting the incumbent via the posterior mean in BO. They find that reshuffling improves generalization – especially with holdout as resampling – and that posterior mean selection can further enhance performance. In contrast, additionally holding out a separate selection set harms generalization. While these results support the effectiveness of these strategies, overtuning itself is not formally quantified – its presence is implicitly inferred from improvements in generalization. Nagler et al. (2024) extend this work by demonstrating that reshuffling improves generalization even for a simple RS, analyzing its effect on the validation loss surface and deriving regret bounds in the asymptotic regime.

Fabris and Freitas (2019) investigate overfitting in the context of AutoML, conducting experiments with Auto-sklearn (Feurer et al., 2015) on 17 datasets using ROC AUC as the performance metric. They analyze discrepancies across three data partitions: training vs. internal validation, training vs. external test, and internal validation vs. external test – the latter aligning with what we term meta-overfitting. Meta-overfitting is prevalent on smaller datasets (1000 observations or fewer). While validation and test scores are generally well-correlated, the number of HPO iterations by SMAC (Hutter et al., 2011) shows no significant correlation with the extent of meta-overfitting.

5 An Empirical Analysis of Overtuning

To evaluate the prevalence and practical significance of overtuning in HPO, we re-analyze several recent, large-scale studies, where the HPO trajectories are publicly available. Specifically, we consider HPO data from the following works: *FCNet* (Klein and Hutter, 2019), *LCBench* (Zimmer et al., 2021), *WDTB* (Grinsztajn et al., 2022), *TabZilla* (McElfresh et al., 2023), *TabRepo* (Salinas and Erickson, 2024), *reshuffling* (Nagler et al., 2024) and *PD1* (Wang et al., 2024). We selected these studies because they include multiple learning algorithms, datasets, and performance metrics. Importantly, each study provides both validation and test performance (estimated on an outer test set), enabling an assessment of overtuning. Each study comprises the evaluation of multiple HPCs for a given combination of learning algorithm, dataset, and performance metric. All studies employed either random search (RS; Bergstra and Bengio 2012) or a fixed grid of HPCs, the latter allowing for simulation of RS. The *reshuffling* study additionally includes BO runs, and runs where the resampling was reshuffled and runs where models were not retrained prior to evaluating on the outer test set, which are excluded from the present analysis and revisited in detail in Section 6.

Our empirical analysis aims to answer the questions: 1) How often does overtuning in HPO occur? 2) How strong is the effect? For each HPO run, defined by a unique tuple of learning algorithm, dataset, performance metric, evaluation protocol, and potentially random seed, we compute the relative overtuning as defined in Definition (3.2). Note that the denominator in Equation (7) can cause numerical instabilities. If the default HPC achieves the best test performance over all incumbents or the improvement is small, the denominator will be zero or close to zero, resulting in the fraction approaching infinity or being undefined. Therefore, when quantifying the overtuning effect at a time point t, it is reasonable to only consider and average over HPO runs where some improvement over the default can be observed with respect to test performance. We use a threshold of $\epsilon = 0.001$ (with the scale of metrics for, e.g., accuracy and ROC AUC ranging from 0 to 1). This procedure yields a distribution of relative overtuning values per study. Approximately 38.5% of HPO runs yield test performance improvements smaller than this threshold.

We visualize the empirical cumulative distribution function (ECDF) over these values in Figure 2 (solid black line). The analysis reveals that in approximately 60% of HPO runs, no overtuning is observed. Furthermore, 70% of runs exhibit relative overtuning less than 0.1, while 90% remain below 1.0. Conversely, this implies that in 10% of HPO runs, we observe severe overtuning (i.e., relative overtuning greater than 1.0). Due to the large variation in the number of HPO runs across studies, we also provide per-study ECDFs in Figure 2. These show substantial heterogeneity: some studies, such as *FCNet*, display almost no overtuning, whereas others, notably *reshuffling* and *TabRepo*, exhibit overtuning in over 50% of runs and severe overtuning in more than 15%. Additional ECDFs stratified by learning algorithm, performance metric, and evaluation protocol for each study are provided in Appendix D. A brief summary of key findings is presented below. For *reshuffling* (Figure 3) we observe that across all learning algorithms and performance metrics, overtuning is substantially mitigated when using 5x 5-fold CV, compared to a simple holdout. HPO based on accuracy and ROC AUC tend to result in higher overtuning, whereas log loss is generally more robust. Among the learning algorithms, the Elastic Net displays the lowest sensitivity to overtuning. In contrast, more flexible models as the Funnel MLP, XGBoost and especially CatBoost show substantial overtuning under holdout, although this can be largely alleviated with more sophisticated resamplings. For *WDTB* (Figure 4), we observe that overtuning is most pronounced for classification tasks evaluated using accuracy, particularly on the categorical classification benchmarks. In contrast, numerical regression tasks using R^2 exhibit substantially lower overtuning. Among learning algorithms, tree-based models such as GradientBoostingTree and HistGradientBoostingTree demonstrate the greatest robustness. Neural architectures, particularly the ResNet and MLP show higher overtuning, especially on classification tasks. Looking at *TabZilla* (Figures 5, 6 7, 8), we observe that the tree-based gradient boosting algorithms are relatively robust

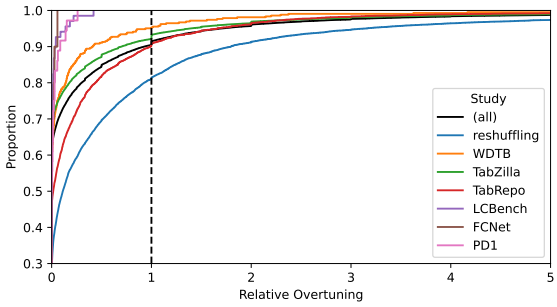


Figure 2: ECDFs of relative overtuning over different HPO studies. y-axis starts at 0.3.

to overtuning, particularly on multiclass classification. Neural architectures including the ResNet and especially the MLPs are more prone to overtuning. In general, binary classification (ROC AUC) is more sensitive to overtuning than multiclass classification (log loss). For TabRepo (Figure 9), we observe the similar trend that binary classification (ROC AUC) is more sensitive to overtuning than multiclass classification (log loss) or regression (RMSE). Moreover, CatBoost and the two neural architectures are more prone to overtuning than the other learning algorithms. For LCBench (Figure 10), PD1 (Figure 11) and FCNet (Figure 12) we observe minimal overtuning but notice that accuracy or classification error are more sensitive to overtuning than cross-entropy, i.e. log loss.

6 Modeling the Determinants of Overtuning

To directly investigate overtuning in HPO and identify influential factors, such as learning algorithms, performance metrics, evaluation protocols, and optimizers, we analyzed the *reshuffling* HPO data reported in Nagler et al. (2024) in more detail. In that study, the authors systematically varied the learning algorithm (Elastic Net, Funnel MLP, XGBoost, CatBoost), performance metric (accuracy, log loss, ROC AUC), dataset size ($n = 500, 1000$, or 5000 observations, with a fixed outer test set of size 5000), and resampling method (80/20 holdout, 5-fold CV, 5x 80/20 holdout, 5x 5-fold CV) in a full factorial design, repeating each HPO run on ten binary classification datasets (treated as data generating processes) ten times. For our analysis, we focus on results from RS with a budget of 500 HPC evaluations under default non-reshuffled resampling, comprising 14400 HPO runs in total. Test performance was determined by retraining the inducer configured by a given HPC on all data and evaluating on the outer holdout set. For more details, see Nagler et al. (2024).

We investigate how overtuning is influenced by the number of HPO iterations, performance metric, learning algorithm (classifier), resampling method, and dataset size. Rather than testing strict hypotheses, our analysis is exploratory (Herrmann et al., 2024). Overtuning and relative overtuning are computed per HPO run as defined in Definition (3.1). Since many runs show no overtuning, we first fit a generalized linear mixed-effects model (GLMM) to predict the probability of nonzero overtuning. The model includes random intercepts for dataset and seed, and fixed effects for the performance metric, classifier, resampling method, dataset size, and a scaled HPO budget (0 to 1), including a quadratic term for the budget to capture nonlinearity. We omit interaction terms to keep the model simple. Results are shown in Table 1a. We observe that longer tuning increases the odds of overtuning (positive main effect), but the negative quadratic term indicates a diminishing effect at higher iteration counts, forming a plateau similar to an inverted U-shape. A likelihood ratio test confirmed the necessity of the quadratic term ($\chi^2(1) = 1913.90, p < 0.001$). Compared to the reference levels (accuracy for metric and Elastic Net for classifier), both log loss and ROC AUC increase the odds of overtuning, and all classifiers increase these odds. In contrast, employing more sophisticated resampling methods (especially 5x 5-fold CV compared to holdout) and using more data ($n = 1000$ or $n = 5000$ observations instead of $n = 500$) reduces the odds.

(a) Fixed effects results from a GLMM predicting probability of nonzero overtuning.

Predictor	Estimate	Std. Error	z value	p-value
(intercept)	-1.408569	0.096734	-14.560	< 0.001
budget	2.262315	0.035965	62.900	< 0.001
budget^2	-1.495099	0.034100	-43.840	< 0.001
metric (ROC AUC)	0.724608	0.006277	115.440	< 0.001
metric (log loss)	0.222283	0.006186	35.930	< 0.001
classifier (CatBoost)	1.609866	0.007494	214.810	< 0.001
classifier (Funnel MLP)	1.200907	0.007357	163.230	< 0.001
classifier (XGBoost)	1.336699	0.007390	180.890	< 0.001
resampling (5x Holdout)	-0.264233	0.007202	-36.690	< 0.001
resampling (5-fold CV)	-0.290060	0.007202	-40.270	< 0.001
resampling (5x 5-fold CV)	-0.481657	0.007214	-66.770	< 0.001
dataset size (1000)	-0.275075	0.006229	-44.160	< 0.001
dataset size (5000)	-0.640027	0.006261	-102.230	< 0.001

(b) Fixed effects results from an LMM predicting nonzero relative overtuning on log scale.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	-1.962e+00	1.465e-01	1.809e+01	-13.389	< 0.001
budget	3.955e-01	4.089e-02	2.616e+05	9.672	< 0.001
budget^2	-2.093e-01	3.737e-02	2.616e+05	-5.600	< 0.001
metric (ROC AUC)	-2.178e-01	6.922e-03	2.616e+05	-31.463	< 0.001
metric (log loss)	-7.852e-01	7.167e-03	2.616e+05	-109.548	< 0.001
classifier (CatBoost)	2.693e+00	8.887e-03	2.616e+05	302.994	< 0.001
classifier (Funnel MLP)	1.218e+00	8.855e-03	2.616e+05	137.563	< 0.001
classifier (XGBoost)	2.176e+00	9.235e-03	2.616e+05	235.615	< 0.001
resampling (5x Holdout)	-3.165e-01	7.390e-03	2.616e+05	-42.831	< 0.001
resampling (5-fold CV)	-3.081e-01	7.371e-03	2.616e+05	-41.793	< 0.001
resampling (5x 5-fold CV)	-4.927e-01	7.530e-03	2.616e+05	-65.437	< 0.001
dataset size (1000)	-1.291e-01	6.285e-03	2.616e+05	-20.549	< 0.001
dataset size (5000)	-4.136e-01	6.658e-03	2.616e+05	-62.129	< 0.001

Table 1: Fixed effects results of mixed models used to analyze overtuning. RS runs, no reshuffling, test performance of the model retrained on all data. Reference levels: accuracy (metric), Elastic Net (classifier), holdout (resampling), 500 (dataset size).

(a) Fixed effects results from an LMM predicting final meta-overfitting.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	3.016e-02	3.803e-03	1.076e+01	7.931	< 0.001
metric (ROC AUC)	2.148e-02	7.691e-04	1.437e+04	27.930	< 0.001
metric (log loss)	-3.619e-03	7.691e-04	1.437e+04	-4.705	< 0.001
classifier (CatBoost)	2.064e-02	8.880e-04	1.437e+04	23.242	< 0.001
classifier (Funnel MLP)	1.736e-02	8.880e-04	1.437e+04	19.545	< 0.001
classifier (XGBoost)	1.154e-02	8.880e-04	1.437e+04	12.998	< 0.001
resampling (5x Holdout)	-1.733e-02	8.880e-04	1.437e+04	-19.514	< 0.001
resampling (5-fold CV)	-2.022e-02	8.880e-04	1.437e+04	-22.769	< 0.001
resampling (5x 5-fold CV)	-2.830e-02	8.880e-04	1.437e+04	-31.868	< 0.001
dataset size (1000)	-1.281e-02	7.691e-04	1.437e+04	-16.661	< 0.001
dataset size (5000)	-2.562e-02	7.691e-04	1.437e+04	-33.317	< 0.001

(b) Fixed effects results from an LMM predicting final test regret.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	1.082e-02	1.333e-03	1.305e+01	8.120	< 0.001
metric (ROC AUC)	1.154e-02	4.157e-04	1.437e+04	27.754	< 0.001
metric (log loss)	-1.240e-04	4.157e-04	1.437e+04	-0.298	< 0.001
classifier (CatBoost)	6.822e-03	4.800e-04	1.437e+04	14.212	< 0.001
classifier (Funnel MLP)	1.215e-02	4.800e-04	1.437e+04	25.321	< 0.001
classifier (XGBoost)	4.122e-03	4.800e-04	1.437e+04	8.587	< 0.001
resampling (5x Holdout)	-5.437e-03	4.800e-04	1.437e+04	-11.327	< 0.001
resampling (5-fold CV)	-5.839e-03	4.800e-04	1.437e+04	-12.164	< 0.001
resampling (5x 5-fold CV)	-7.362e-03	4.800e-04	1.437e+04	-15.338	< 0.001
dataset size (1000)	-5.352e-03	4.157e-04	1.437e+04	-12.876	< 0.001
dataset size (5000)	-1.027e-02	4.157e-04	1.437e+04	-24.696	< 0.001

Table 2: Fixed effects results of mixed models used to analyze final meta-overfitting and test regret. RS runs, no reshuffling, test performance of the model retrained on all data. Reference levels of factors are: accuracy (metric), Elastic Net (classifier), holdout (resampling), 500 (dataset size).

As a follow up, we fitted a linear mixed-effects model (LMM) to predict the relative overtuning as in Definition (3.2) on a logarithmic scale (to counter skewness) for cases with nonzero overtuning. The LMM uses the same random and fixed effects as the GLMM, and a likelihood ratio test again confirmed the need for a quadratic budget term ($\chi^2(1) = 31.361, p < 0.001$). Table 1b summarizes these results. Conclusions largely remain the same as for the GLMM, i.e., using a more sophisticated resampling method and more data reduces the extent of overtuning although ROC AUC and log loss now overall show less nonzero relative overtuning compared to accuracy. As before, we observe that longer tuning increases relative overtuning (positive main effect), but the negative quadratic term indicates a diminishing effect. Finally, we fitted LLMs to predict the final meta-overfitting and final test regret after 500 HPO iterations. Results in Table 2a and Table 2b show that employing more sophisticated resampling methods and using larger datasets reduce both meta-overfitting and test regret. These findings suggest that practitioners should prefer CV (repeated if possible) over holdout validation whenever possible, particularly with small datasets.

To assess the effect of the optimizer (RS vs. HEBO, see Cowen-Rivers et al. 2022 vs. SMAC3, see Lindauer et al. 2022), we conducted another mixed model analysis on the reshuffling data subset, limited to 250 iterations (the BO budget), using ROC AUC as the performance metric (the only one tracked in BO experiments). The choice of optimizer was included as a fixed effect and other random and fixed effects remained the same as in the previous modeling approach. A likelihood ratio test revealed a significant effect of the optimizer for both the GLMM modeling the probability of nonzero overtuning and the LMM modeling the nonzero relative overtuning on log scale: $\chi^2(2) = 416.14, p < 0.001$ for the GLMM, and $\chi^2(2) = 1509.7, p < 0.001$ for the LMM. In the GLMM (Table 3a), we observed small but significant positive coefficients for both HEBO (0.0833, $z = 9.049$, $p < 0.001$) and SMAC3 (0.1881, $z = 20.347$, $p < 0.001$), compared to RS, suggesting that both BO methods slightly increase the odds of nonzero overtuning. Conversely, the LMM analysis of the

magnitude of overtuning (Table 3b) showed significant negative coefficients for HEBO (-0.3011, $t(154200) = -36.363$, $p < 0.001$) and SMAC (-0.2581, $t(154200) = -30.956$, $p < 0.001$), indicating that while BO slightly increases the likelihood of any overtuning, it substantially reduces its magnitude compared to RS. Finally, based on an LMM modeling final test regret with optimizer as a fixed factor (Table 4b), we found that HEBO significantly reduces test regret relative to RS (-0.0021, $t(14370) = -5.167$, $p < 0.001$), suggesting that HEBO tends to identify HPCs that generalize better. SMAC3 also showed a small negative coefficient (-0.0003, $t(14370) = -0.691$, $p = 0.489$), but this effect is not statistically significant.

We also investigated the effect of early stopping in BO (Makarova et al., 2021, 2022) by comparing HEBO with HEBO using early stopping on the data subset up to 250 iterations (the BO budget), using 5-fold CV as the resampling method (the only setting where early stopping à la Makarova et al. (2022) is directly applicable), and ROC AUC as the performance metric (the only one tracked in BO experiments). We applied the same mixed model analysis framework as before. Likelihood ratio tests revealed a significant effect of early stopping for both the probability of nonzero overtuning (GLMM: $\chi^2(1) = 36.077$, $p < 0.001$) and the extent of nonzero relative overtuning on log scale (LMM: $\chi^2(1) = 10.720$, $p = 0.001$). When including early stopping as a fixed factor (Table 5b), we observed a negative coefficient (-0.27531, $t(980.555) = -3.272$, $p = 0.001$) for nonzero relative overtuning on log scale indicating a mitigating effect, albeit comparably small. One reason can be that this analysis is restricted to HPO runs using 5-fold CV, where we have seen that overtuning is rather mild, when compared to holdout runs. Moreover, since HEBO already reduces overtuning compared to RS, the additional benefit from applying early stopping can be rather incremental.

Last but not least, we turn to the core idea behind the reshuffling data: reshuffling the resampling splits during HPO, a strategy shown to improve generalization performance, particularly in the case of holdout resampling (Nagler et al., 2024). We conducted a mixed model analysis as before on the reshuffling data, focusing on the larger subset of RS runs (500 iterations). A likelihood ratio test indicated a significant effect of reshuffling for both the GLMM modeling the probability of nonzero overtuning ($\chi^2(1) = 152.54$, $p < 0.001$) and the LMM modeling the nonzero relative overtuning on log scale ($\chi^2(1) = 181.10$, $p < 0.001$). Specifically, we find that, overall, reshuffling slightly increases the odds of overtuning (0.0439, $z = 12.351$, $p < 0.001$, Table 6a) as well as its extent (0.0515, $t(537900) = 13.458$, $p < 0.001$, Table 6b). Nagler et al. (2024) demonstrated that reshuffling can improve generalization especially when holdout is used as a resampling with ROC AUC as the performance metric. When we restrict our analysis to this particular setting, we observe a clear shift. For both the GLMM (Table 8a) and LMM (Table 8b), reshuffling has a significant negative effect on overtuning: it strongly decreases the odds of overtuning (-0.2645, $z = -20.054$, $p < 0.001$) and its extent (-0.2693, $t(55480) = -23.236$, $p < 0.001$). Moreover, reshuffling significantly reduces final test regret in this setting, as shown by an LMM analysis (-0.0057, $t(2375) = -5.990$, $p < 0.001$, Table 9b), indicating that it leads to the identification of HPCs with better true generalization performance. We find that reshuffling actually increases final meta-overfitting (0.0548, $t(2375) = 25.382$, $p < 0.001$, Table 9a). However, meta-overfitting does not necessarily imply worse HPO generalization. In fact, the "hedging"' effect of reshuffling described by Nagler et al. (2024) appears strong enough to reduce both overtuning and test regret.

7 Mitigation Strategies

While the primary contribution of this paper is to highlight the issue of overtuning in HPO, we now turn to a discussion of potential mitigation strategies, drawing from both Section 6 and existing literature. Broadly, these strategies fall into three categories: 1) Modifying the objective function, 2) adjusting incumbent selection, and 3) modifying the optimizer producing the HPC trajectory. The first includes methods that reduce variance or add regularization. The second includes early stopping and avoiding naïve selection of the validation-optimal HPC. The third is broader and includes any changes to the optimizer that produces the HPC trajectory.

We have seen in our analysis of the data from Nagler et al. (2024) that larger datasets generally reduce both the likelihood and severity of overtuning. While this is expected, we note that increasing dataset size is often infeasible in practice. As such, overtuning remains a primary concern in small-data regimes. Second, more advanced resampling strategies such as CV or repeated CV substantially reduce both the frequency and extent of overtuning. These strategies, along with larger datasets, also help mitigate meta-overfitting and decrease test regret, enabling HPO to more reliably identify configurations that generalize well. Third, our findings suggest that BO results in less overtuning than RS, although it may slightly increase meta-overfitting. This trade-off deserves further study. One possible explanation is that BO more effectively identifies configurations with exceptionally strong validation performance that also generalize well. Additionally, BO's use of a surrogate model may help smooth over noise in validation estimates, guiding the search toward more robust regions. In noisy BO, one can select the incumbent based on the surrogate's posterior predictive distribution rather than the empirically best configuration (Picheny et al., 2013). While this was not implemented in the BO runs of Nagler et al. (2024), incorporating such noise-aware techniques may further reduce overtuning as briefly touched upon by Lévesque (2018). Finally, reshuffling resampling splits as done in Lévesque (2018); Nagler et al. (2024) can help mitigate overtuning, although its effectiveness varies across performance metrics, algorithms, and resampling methods.

Prior work has touched on several strategies to mitigate overtuning. Cawley and Talbot (2010) briefly mention regularization, early stopping, and model averaging. For example, Cawley and Talbot (2007) show that incorporating L2 regularization on lengthscale parameters in kernel methods can improve generalization. However, in modern tabular learning settings, applying regularization during HPO is challenging, as it requires a clear mapping between hyperparameters and model complexity – something not always available. Early stopping, explored by Makarova et al. (2021, 2022), shows promise, but in our analysis, it did not strongly reduce overtuning. One reason could be that stopping too early may prevent discovering genuinely better configurations. Another issue lies in the reliability of the variance estimator used for the stopping criterion, where we know that no unbiased variance estimator exists for CV performance estimates (Bengio and Grandvalet, 2004). Finally, a fully Bayesian treatment of hyperparameters, as presented in Williams and Barber (1998) and discussed by Cawley and Talbot (2010), appears impractical for modern models due to computational and modeling complexity.

Other mitigation strategies focus on more cautious incumbent selection, such as using a dedicated selection set or applying conservative selection criteria. Several works (Dos Santos et al., 2009; Koch et al., 2010; Igel, 2012; Lévesque, 2018) have explored selecting the final HPC based on a separate test set. However, Lévesque (2018) report that a dedicated test set can degrade generalization performance, as it reduces the data available for HPO. Similarly, ML-Plan (Mohr et al., 2018) adopts a two-phase strategy: It first explores candidates using one data split and then selects the most robust model via conservative GE estimates on a held-out subset. LOOCVCV (Ng, 1997) proposes selecting the incumbent not with the best validation error, but at an adaptively chosen percentile, based on how many configurations can be evaluated before overtuning occurs. While effective in noisy settings, it tends to be overly conservative in low-noise regimes and is limited to decomposable pointwise metrics and i.i.d. configurations, restricting it to RS while adding computational overhead.

We have seen that an effective and simple mitigation strategy against overtuning is using more robust resampling methods like repeated CV. However, this comes with increased computational cost. To balance robustness and efficiency, it may be worthwhile to revisit adaptive resampling techniques (Thornton et al., 2013; Zheng and Bilenko, 2013; Bergman et al., 2024), racing (Birattari et al., 2002; Lang et al., 2015) or optimal budget allocation strategies (Bartz-Beielstein et al., 2011).

8 Broader Impact Statement

This work presents no notable negative impacts to society or the environment.

References

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79.

Arora, S. and Zhang, Y. (2021). Rip van Winkle's razor: A simple estimate of overfit to test data.

Austern, M. and Zhou, W. (2020). Asymptotics of cross-validation.

Barros, R. C., de Carvalho, A. C. P. L. F., and Freitas, A. A. (2015). *Automatic Design of Decision-Tree Induction Algorithms*. Springer International Publishing, Cham.

Bartz-Beielstein, T., Friese, M., Zaefferer, M., Naujoks, B., Flasch, O., Konen, W., and Koch, P. (2011). Noisy optimization with sequential parameter optimization and optimal computational budget allocation. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, page 119–120.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1—-48.

Bates, S., Hastie, T., and Tibshirani, R. (2024). Cross-validation: what does it estimate and how well does it do it? *Journal of the American Statistical Association*, 119(546):1434–1445.

Bayle, P., Bayle, A., Janson, L., and Mackey, L. (2020). Cross-validation confidence intervals for test error. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H., editors, *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, pages 16339–16350. Curran Associates.

Bengio, Y. and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 4:1089–1105.

Bergman, E., Purucker, L., and Hutter, F. (2024). Don't waste your time: Early stopping cross-validation. In Lindauer et al. (2024), pages 9/1–31.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In Langdon, W., Cantu-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M., Schultz, A., Miller, J., Burke, E., and Jonoska, N., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)*, pages 11–18. Morgan Kaufmann Publishers.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., and Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1484.

Blum, A. and Hardt, M. (2015). The ladder: A reliable leaderboard for machine learning competitions. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, volume 37, pages 1006–1014. Omnipress.

Breiman, L. (1984). *Classification and regression trees*. Routledge.

Cawley, G. and Talbot, N. (2010). On Overfitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, 11:2079–2107.

Cawley, G. C. and Talbot, N. L. C. (2007). Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8(31):841–861.

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., and Le, Q. V. (2023). Symbolic discovery of optimization algorithms. In Oh et al. (2023).

Cowen-Rivers, A., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R., Maraval, A., Jianye, H., Wang, J., Peters, J., and Ammar, H. (2022). HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349.

Dos Santos, E. M., Sabourin, R., and Maupin, P. (2009). Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, 10(2):150–162.

Dunias, Z. S., van Calster, B., Timmerman, D., Boulesteix, A.-L., and van Smeden, M. (2024). A comparison of hyperparameter tuning procedures for clinical prediction models: A simulation study. *Statistics in Medicine*, 43(6):1119–1134.

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. (2015). Generalization in adaptive data analysis and holdout reuse. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The 632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

Eggensperger, K., Lindauer, M., and Hutter, F. (2019). Pitfalls and best practices in algorithm configuration. *Journal of Artificial Intelligence Research*, pages 861–893.

Eimer, T., Lindauer, M., and Raileanu, R. (2023). Hyperparameters in reinforcement learning and how to tune them. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, volume 202 of *Proceedings of Machine Learning Research*. PMLR.

Escalante, H., Montes, M., and Sucar, E. (2009). Particle Swarm Model Selection. *Journal of Machine Learning Research*, 10:405–440.

Fabris, F. and Freitas, A. (2019). Analysing the overfit of the auto-sklearn automated machine learning tool. In Nicosia, G., Pardalos, P., Umeton, R., Giuffrida, G., and Sciacca, V., editors, *Machine Learning, Optimization, and Data Science*, volume 11943 of *Lecture Notes in Computer Science*, pages 508–520.

Feldman, V., Frostig, R., and Hardt, M. (2019). The advantages of multiple classes for reducing overfitting from test set reuse. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, volume 97, pages 1892–1900. Proceedings of Machine Learning Research.

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. (2022). Auto-Sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23(261):1–61.

Feurer, M. and Hutter, F. (2019). Hyperparameter Optimization. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Automated Machine Learning: Methods, Systems, Challenges*, chapter 1, pages 3 – 38. Springer. Available for free at `http://automl.org/book`.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Proceedings of the 29th International Conference on Advances in Neural Information Processing Systems (NeurIPS'15)*, pages 2962–2970. Curran Associates.

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press. Available for free at `https://bayesoptbook.com/`.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Proceedings of the 35th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates.

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macià, N., Ray, B., Saeed, M., Statnikov, A., and Viegas, E. (2015). Design of the 2015 ChaLearn AutoML challenge. In *2015 International Joint Conference on Neural Networks (IJCNN'15)*, pages 1–8. International Neural Network Society and IEEE Computational Intelligence Society, IEEE.

Guyon, I., Saffari, A., Dror, G., and Cawley, G. (2010). Model selection: Beyond the Bayesian/Frequentist divide. *Journal of Machine Learning Research*, 11:61–87.

Hardt, M. (2017). Climbing a shaky ladder: Better adaptive risk estimation.

Herrmann, M., Lange, F., Eggensperger, K., Casalicchio, G., Wever, M., Feurer, M., Rügamer, D., Hüllermeier, E., Boulesteix, A.-L., and Bischl, B. (2024). Position: Why we must rethink empirical research in machine learning. In Salakhutdinov et al. (2024), pages 18228–18247.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. In Lee, K. M., editor, *IEEE Transactions on Pattern Analysis and Machine Intelligence'21)*. IEEE Computer Society.

Huisman, M., van Rijn, J., and Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, 54:4483–4541.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello, C., editor, *Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION'11)*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer.

Igel, C. (2012). A note on generalization loss when evolving adaptive pattern recognition systems. *IEEE Transactions on Evolutionary Computation*, 17(3):345–352.

Ishibashi, H., Karasuyama, M., Takeuchi, I., and Hino, H. (2023). A stopping criterion for Bayesian optimization by the gap of expected minimum simple regrets. In Ruiz, F., Dy, J., and van de Meent, J.-W., editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pages 6463–6497.

Klein, A. and Hutter, F. (2019). Tabular benchmarks for Joint Architecture and Hyperparameter optimization. *arXiv:1905.04970[cs.LG]*.

Koch, P., Konen, W., Flasch, O., and Bartz-Beielstein, T. (2010). Optimizing support vector machines for stormwater prediction. Technical Report TR10-2-007, Technische Universität Dortmund. Proceedings of Workshop on Experimental Methods for the Assessment of Computational Systems joint to PPSN2010.

Lang, M., Kotthaus, H., Marwedel, P., Weihs, C., Rahnenführer, J., and Bischl, B. (2015). Automatic model selection for high-dimensional survival analysis. *Journal of Statistical Computation and Simulation*, 85:62–76.

Larcher, C. and Barbosa, H. (2022). Evaluating models with dynamic sampling holdout in auto-ml. *SN Computer Science*, 3(506).

Li, S., Li, K., and Li, W. (2023). "Why not looking backward?" A robust two-step method to automatically terminate Bayesian optimization. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 43435–43446.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A versatile bayesian optimization package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Lindauer, M., Eggensperger, K., Garnett, R., Vanschoren, J., and Gardner, J., editors (2024). *Proceedings of the Third International Conference on Automated Machine Learning*. Proceedings of Machine Learning Research.

Lorenz, R., Monti, R. P., Violante, I. R., Faisal, A. A., Anagnostopoulos, C., Leech, R., and Montana, G. (2016). Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization.

Loughrey, J. and Cunningham, P. (2005). Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In Bramer, M., Coenen, F., and Allen, T., editors, *Research and Development in Intelligent Systems XXI*, pages 33–43, London. Springer London.

Loya, H., Łukasz Dudziak, Mehrotra, A., Lee, R., Fernandez-Marques, J., Lane, N. D., and Wen, H. (2023). How much is hidden in the NAS benchmarks? few-shot adaptation of a NAS predictor. *arXiv:2311.18451 [cs.LG]*.

Lévesque, J. (2018). *Bayesian Hyperparameter Optimization: Overfitting, Ensembles and Conditional Spaces*. PhD thesis, Université Laval.

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J., Krause, A., Seeger, M., and Archambeau, C. (2021). Overfitting in Bayesian Optimization: An empirical study and early-stopping solution. In *ICLR 2021 Workshop on Neural Architecture Search*.

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J., Krause, A., Seeger, M., and Archambeau, C. (2022). Automatic termination for hyperparameter optimization. In Guyon, I., Lindauer, M., van der Schaar, M., Hutter, F., and Garnett, R., editors, *Proceedings of the First International Conference on Automated Machine Learning*. Proceedings of Machine Learning Research.

McCulloch, C. E., Searle, S. R., and Neuhaus, J. M. (2008). *Generalized, Linear, and Mixed Models*. Wiley Series in Probability and Statistics. Wiley, Hoboken, NJ, 2 edition.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. (2023). When do neural nets outperform boosted trees on tabular data? In Oh et al. (2023), pages 76336–76369.

Mohr, F., Wever, M., and Hüllermeier, E. (2018). ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8-10):1495–1515.

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: A comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307.

Nagler, T., Schneider, L., Bischl, B., and Feurer, M. (2024). Reshuffling resampling splits can improve generalization of hyperparameter optimization. In *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'24)*. Curran Associates.

Neto, E. C., Hoff, B. R., Bare, C., Bot, B. M., Yu, T., Magravite, L., Trister, A. D., Norman, T., Meyer, P., Saez-Rodrigues, J., Costello, J. C., Guinney, J., and Stolovitzky, G. (2016). Reducing overfitting in challenge-based competitions.

Ng, A. (1997). Preventing "overfitting"' of cross-validation data. In Fisher, D., editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pages 245–253. Morgan Kaufmann Publishers.

Nguyen, T., Gupta, S., Rana, S., and Venkatesh, S. (2018). Stable bayesian optimization. *International Journal of Data Science and Analytics*, 6:327–339.

Nguyen, V., Gupta, S., Rana, S., Li, C., and Venkatesh, S. (2017). Regret for expected improvement over the best-observed value and stopping condition. In Zhang, M.-L. and Noh, Y.-K., editors, *Proceedings of the Ninth Asian Conference on Machine Learning*, volume 77, pages 279–294.

Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors (2023). *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*. Curran Associates.

Paraschakis, K., Castellani, A., Borboudakis, G., and Tsamardinos, I. (2024). Confidence interval estimation of predictive performance in the context of AutoML. In Eggensperger, K., Garnett, R., Vanschoren, J., Lindauer, M., and Gardner, J. R., editors, *Proceedings of the Third International Conference on Automated Machine Learning*, volume 256, pages 4/1–14.

Picheny, V., Wagner, T., and Ginsbourger, D. (2013). A benchmark of Kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48:607–626.

Probst, P., Boulesteix, A., and Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(53):1–32.

Quinlan, J. and Cameron-Jones, R. (1995). Oversearching and layered search in empirical learning. In Mellish, C., editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, page 1019–1024. Morgan Kaufmann Publishers.

Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3:1371–1382.

Riley, R. D., Snell, K. I. E., Martin, G. P., Whittle, R., Archer, L., Sperrin, M., and Collins, G. S. (2021). Penalization and shrinkage methods produced unreliable clinical prediction models especially when sample size was small. *Journal of Clinical Epidemiology*, 132:88–96.

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., and Schmidt, L. (2019). A meta-analysis of overfitting in machine learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alche Buc, F., Fox, E., and Garnett, R., editors, *Proceedings of the 32nd International Conference on Advances in Neural Information Processing Systems (NeurIPS'19)*. Curran Associates.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors (2024). *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, volume 251 of *Proceedings of Machine Learning Research*. PMLR.

Salinas, D. and Erickson, N. (2024). TabRepo: A large scale repository of tabular model evaluations and its AutoML applications. In Lindauer et al. (2024), pages 19/1–30.

Schulz-Kümpel, H., Fischer, S., Hornung, R., Boulesteix, A.-L., Nagler, T., and Bischl, B. (2025). Constructing confidence intervals for 'the' generalization error – a comprehensive benchmark study.

Song, X., Tian, Y., Lange, R. T., Lee, C., Tang, Y., and Chen, Y. (2024). Position: Leverage foundational models for black-box optimization. In Salakhutdinov et al. (2024), pages 46168–46180.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133.

Thornton, C., Hutter, F., Hoos, H., and Leyton-Brown, K. (2013). Auto-WEKA: combined selection and Hyperparameter Optimization of classification algorithms. In Dhillon, I., Koren, Y., Ghani, R., Senator, T., Bradley, P., Parekh, R., He, J., Grossman, R., and Uthurusamy, R., editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, pages 847–855. ACM Press.

Van Calster, B., van Smeden, M., De Cock, B., and Steyerberg, E. W. (2020). Regression shrinkage methods for clinical prediction models do not guarantee improved performance: Simulation study. *Statistical Methods in Medical Research*, 29(11):3166–3178.

van Rijn, J. and Hutter, F. (2018). Hyperparameter importance across datasets. In Guo, Y. and Farooq, F., editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*, pages 2367–2376. ACM Press.

Wainer, J. and Cawley, G. (2017). Empirical Evaluation of Resampling Procedures for Optimising SVM Hyperparameters. *Journal of Machine Learning Research*, 18:1–35.

Wang, Z., Dahl, G. E., Swersky, K., Lee, C., Nado, Z., Gilmer, J., Snoek, J., and Ghahramani, Z. (2024). Pre-trained Gaussian processes for Bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):13420–1351.

Wilson, J. T. (2024). Stopping Bayesian optimization with probabilistic regret bounds. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 98264–98296.

Yang, C., Akimoto, J., Kim, D., and Udell, M. (2019). OBOE: Collaborative filtering for AutoML model selection. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., and Karypis, G., editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*, pages 1173–1183. ACM Press.

Yang, C., Fan, J., Wu, Z., and Udell, M. (2020). AutoML pipeline selection: Efficiently navigating the combinatorial space. In Tang, J. and Prakash, B., editors, *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20)*. ACM Press.

Yao, H., Huang, L.-K., Zhang, L., Wei, Y., Tian, L., Zou, J., Huang, J., and Li, Z. . (2021). Improving generalization in meta-learning via task augmentation. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, volume 139 of *Proceedings of Machine Learning Research*, pages 11887–11897. PMLR.

Zheng, A. and Bilenko, M. (2013). Lazy paired hyper-parameter tuning. In Rossi, F., editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 1924–1931.

Zimmer, L., Lindauer, M., and Hutter, F. (2021). Auto-Pytorch: Multi-fidelity metalearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3079–3090.

Šinkovec, H., Heinze, G., Blagus, R., and Geroldinger, A. (2021). To tune or not to tune, a case study of ridge logistic regression in small or sparse datasets. *BMC Medical Research Methodology*, 21(1):199.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] [Definitions are provided in Section 3, the empirical analysis in Section 5, the modeling of determinants in Section 6 and mitigation strategies are discussed in Section 7].

   (b) Did you describe the limitations of your work? [Yes] [As this paper does not introduce a new algorithm or method but instead focuses on establishing formal definitions and reanalyzing existing HPO studies, we did not include a dedicated limitation section but instead discuss limitations directly where applicable, e.g., in Section 6 we state that we did not include interaction effects in the mixed models. Regarding our definition of overtuning, we discuss limitations in Appendix A.]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] [See Section 8].

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes] [The paper conforms to them.]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [N/A] [We rely on data of various, published works that conducted HPO runs and published this data. We do not compare methods.]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [N/A] [We rely on data of various, published works that conducted HPO runs and published this data. Our analyses of this data does not require data splits, pre-processing, or hyperparameter tuning]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [N/A] [We rely on data of various, published works that conducted HPO runs and published this data. With the exception of additional runs for Section 6 we did not run any experiments but relied on the experiments of existing published work.]

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] [Mixed model analyses include standard error estimates of coefficients.]

   (e) Did you report the statistical significance of your results? [Yes] [Mixed model analyses include measures of statistical significance.]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A] [We rely on data of various, published works that conducted HPO runs and published this data.]

   (g) Did you compare performance over time and describe how you selected the maximum duration? [N/A] [We perform an analysis of the overtuning found in HPO runs of various published works. The authors of these works determined the overall HPO budget which we cannot influence in hindsight. In Section 6 we model overtuning as an anytime metric.]

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] [See Appendix F.]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [N/A] [We rely on data of various, published works that conducted HPO runs and published this data. We do not introduce a new algorithm or method.]

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] [See Appendix F.]

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] [See Appendix F.]

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] [See Appendix F.]

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] [See Appendix F.]

   (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] [See Appendix F.]

4. If you used existing assets (e.g., code, data, models)...

   (a) Did you cite the creators of used assets? [Yes] [We cite the creators of used assets where applicable in Section 5 and Section 6.]

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] [Used existing assets are leased under licenses that permit usage.]

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] [Used existing assets do not contain such information or content.]

5. If you created/released new assets (e.g., code, data, models)...

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] [See Appendix F.]

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] [See Appendix F.]

6. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] [We did not use crowdsourcing or conducted research with human subjects.]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] [We did not use crowdsourcing or conducted research with human subjects.]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] [We did not use crowdsourcing or conducted research with human subjects.]

7. If you included theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] [We do not include theoretical results.]

(b) Did you include complete proofs of all theoretical results? [N/A] [We do not include theoretical results.]

A Limitations

Overtuning and relative overtuning, as defined in Definition 3.1 and Definition 3.2, quantify how much better HPO could have performed if no decisions had been made based on misleading validation error throughout the trajectory of evaluated HPCs. However, these metrics are not designed to assess or compare the absolute generalization performance of different HPO protocols. While this may seem evident, we state it explicitly for clarity. Consider two hypothetical HPO protocols:

- Protocol A evaluates only a single configuration, $\boldsymbol{\lambda}_{\text{A},1} = \boldsymbol{\lambda}^*_{\text{A},1}$, achieving $\widehat{\text{val}}(\boldsymbol{\lambda}^*_{\text{A},1}) = 0.3$ and $\text{test}(\boldsymbol{\lambda}^*_{\text{A},1}) = 0.35$.
- Protocol B in contrast evaluates $t = 10$ configurations, with the final incumbent $\boldsymbol{\lambda}^*_{\text{B},10}$ achieving $\widehat{\text{val}}(\boldsymbol{\lambda}^*_{\text{B},10}) = 0.18$ and $\text{test}(\boldsymbol{\lambda}^*_{\text{B},10}) = 0.22$.

Suppose Protocol B exhibits a final overtuning of $\text{ot}_{10} = 0.02$, implying that an earlier incumbent had a true GE of 0.20. Protocol A, by definition, cannot exhibit overtuning since it only evaluates a single configuration. Nevertheless, Protocol B clearly leads to better generalization performance (0.22 vs. 0.35), and should be preferred when the primary concern is generalization – even though it exhibits overtuning. However, the presence of overtuning in Protocol B is still informative, as it indicates that even better generalization performance was theoretically achievable.

Furthermore, relative overtuning (Definition 3.2) can be sensitive to the scale of possible performance improvements. If the test error difference between the default or first evaluated HPC and the best observed incumbent is small, the relative overtuning may appear disproportionately large. This reflects the metric's design – to measure the relative gain missed due to overtuning – but it can lead to inflated values in scenarios where HPO yields only marginal improvements. We do not investigate in this work why such marginal improvements might occur and how improvements on the validation set can generalize to improvements on an outer test set but note that meta-overfitting (Definition 3.1) is a suitable metric to assess this. Possible explanations include low tunability (Probst et al., 2019; van Rijn and Hutter, 2018) of the learning algorithm, or overly constrained search spaces where all configurations perform similarly. In such cases, high relative overtuning may simply reflect the limited room for improvement rather than poor HPO generalization.

A direct practical implication is that overtuning alone is not sufficient to evaluate or compare HPO protocols. When analyzing mitigation strategies for overtuning, it is essential to consider their impact on absolute generalization performance. In specific settings – such as the RS runs in Nagler et al. (2024), where all protocols use the same fixed trajectory of HPCs – trajectory test regret (Definition 3.1) may suffice, as it directly measures how well a protocol identifies a near-optimal configuration within the same trajectory.

However, when HPO protocols differ in their search trajectories – due to early stopping, different optimizers, or resource budgets – we must also compare the final test performance of their incumbents. Only then can we draw reliable conclusions about which protocol performs better overall with respect to generalization. For the HEBO vs. HEBO with early stopping à la Makarova et al. (2022) analyses reported in Section 6, we therefore provide a follow-up analysis concerned with the test performance of the final incumbent in Appendix E.1.

B Overtuning vs. Meta-Overfitting

Proposition B.1. *Given a sequence of HPC evaluations* $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T)$*, if overtuning exists at a time point* t*, i.e.,* $\text{ot}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T) > 0$*, there must exist some non-zero meta-overfitting for the current incumbent* $\boldsymbol{\lambda}_t^*$ *or some previous incumbent* $\boldsymbol{\lambda}_{t'}^*$ $(t' < t)$*, i.e.,* $\text{of}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T) \neq 0$ *or* $\text{of}_{t'}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots \boldsymbol{\lambda}_T) \neq 0$*.*

We can easily see this by contradiction. Assume that overtuning exists at time point t. By Definition (3.1) this means $(\text{test}(\boldsymbol{\lambda}_t^*) - \min_{\boldsymbol{\lambda}_{t'}^* \in \{\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_t^*\}} \text{test}(\boldsymbol{\lambda}_{t'}^*)) > 0$. Since the minimum is strictly less than $\text{test}(\boldsymbol{\lambda}_t^*)$ there must exist a previous incumbent $\boldsymbol{\lambda}_{t'}^*$ such that $\text{test}(\boldsymbol{\lambda}_{t'}^*) < \text{test}(\boldsymbol{\lambda}_t^*)$. By definition of incumbents, we know: $\widehat{\text{val}}(\boldsymbol{\lambda}_t^*) \leq \widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*)$ since $t' < t$. Assume that meta-overfitting is zero for both $\boldsymbol{\lambda}_t^*$ and $\boldsymbol{\lambda}_{t'}^*$, i.e., $\text{test}(\boldsymbol{\lambda}_t^*) = \widehat{\text{val}}(\boldsymbol{\lambda}_t^*)$ and $\text{test}(\boldsymbol{\lambda}_{t'}^*) = \widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*)$. Substituting in $\text{test}(\boldsymbol{\lambda}_{t'}^*) < \text{test}(\boldsymbol{\lambda}_t^*)$ gives $\widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*) = \text{test}(\boldsymbol{\lambda}_{t'}^*) < \text{test}(\boldsymbol{\lambda}_t^*) = \widehat{\text{val}}(\boldsymbol{\lambda}_t^*)$, from which follows $\widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*) < \widehat{\text{val}}(\boldsymbol{\lambda}_t^*)$ contradicting the established relation $\widehat{\text{val}}(\boldsymbol{\lambda}_t^*) \leq \widehat{\text{val}}(\boldsymbol{\lambda}_{t'}^*)$.

Note that non-zero meta-overfitting, however, is not sufficient to observe overtuning. Assume the following performance values of HPCs $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$: $\widehat{\text{val}}(\boldsymbol{\lambda}_1) = 0.3, \widehat{\text{val}}(\boldsymbol{\lambda}_2) = 0.2, \text{test}(\boldsymbol{\lambda}_1) = 0.4, \text{test}(\boldsymbol{\lambda}_2) = 0.35$. We observe meta-overfitting of $\text{of}_1(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = 0.4 - 0.3 = 0.1$ and $\text{of}_2(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = 0.35 - 0.2 = 0.15$. Still, overtuning is zero as neither for $t = 1$ nor $t = 2$ there exists a previous incumbent with better test performance. In this sense, we correctly identified the best HPC performing with respect to true GE. This relationship of meta-overfitting and overtuning is also depicted in Figure 1.

C Extended Related Work

A foundational study by Cawley and Talbot (2010) shows that any model selection criterion (for instance, CV) inherently has both bias and variance because it relies on a finite data sample. As they illustrate with synthetic data, extensive optimization on the validation set can cause the chosen model to excel on validation performance but fail to generalize to unseen test data. Their real-data experiments focus on comparing final configurations chosen by different model-selection schemes (e.g., a single-parameter RBF kernel vs. an ARD kernel for kernel ridge regression). They do not define a formal metric of overfitting for model selection but empirically demonstrate that more flexible setups, such as ARD, can outperform on validation yet yield worse performance on a held-out test set. Their work is best known for emphasizing nested CV (or nested resampling in general) as essential for unbiased performance estimation once model selection is performed.

Ng (1997) propose an approach closely aligned with the idea of overtuning, although their work has not been widely recognized in contemporary AutoML and HPO research. They critique the practice of selecting models solely by their CV (in actuality, simple holdout) performance, noting that the variance of the validation error estimate can skew the posterior distribution of the true GE. To address this, Ng (1997) suggest selecting not the lowest validation error, but rather the hypothesis at the k-th percentile of validation performance, where k is chosen adaptively. This adaptation relies on LOOCVCV: it estimates how many candidate hypotheses can be evaluated before overfitting the validation error. Although effective under high noise and limited samples (e.g., in synthetic classification with decision trees), this LOOCVCV approach can become overly conservative with lower noise, sometimes underperforming simpler selection strategies.

Guyon et al. (2010) further formalize model selection through a multi-level inference framework that brings together Bayesian, frequentist, and hybrid viewpoints (see also Bischl et al. 2023). They underscore the risk of overfitting in hyperparameter selection – citing Cawley and Talbot (2010) – and advocate for bound-based selection, ensemble methods, and other regularization techniques to mitigate it. Likewise, Guyon et al. (2015) view model selection as a bi-level optimization problem, arguing that one must introduce regularization and robust data-splitting practices to avoid overfitting to empirical criteria like the CV error. Auto-sklearn 2.0 (Feurer et al., 2022) demonstrate that it is also possible to meta-learn the model selection criterion rather than treating it as a static heuristic.

Makarova et al. (2022) address the challenge of deciding when to stop BO in HPO by proposing a new termination criterion. This criterion combines a confidence bound on the surrogate model's regret with a variance estimate of the CV estimator. Their rule halts BO once the maximum plausible improvement from the surrogate falls below the standard deviation of the incumbent's validation error. They report that this avoids many unnecessary function evaluations and saves computational resources, at only a small cost in final test performance. While they briefly acknowledge that the discrepancy between validation and test performance can persist, it is attributed mainly to low validation–test correlation.

An earlier workshop version (Makarova et al., 2021) puts stronger emphasis on "overfitting" in BO, showing that in tuning an Elastic Net, XGBoost, and a random forest across 19 datasets, Elastic Net (trained via SGD) performance on the test set often declined after prolonged validation-driven optimization. Their explanation again points to weak validation–test correlations, though they do not discuss deeper causes (dataset traits, algorithms, metrics, or resampling choices). In their analyses, they employ the Relative Test Error Change (RYC) to compare test errors in early-stopped vs. full-budget runs, and the Relative Time Change (RTC) to quantify computational savings. Positive RYC implies that early stopping helped avert overtuning, whereas negative values mean the run was halted prematurely.

Nguyen et al. (2018) also study overfitting in BO-based HPO, focusing on how to detect "stable" solutions. Their notion of stability involves low "extra variance", defined as the change in predictive

mean and variance under small Gaussian perturbations of the hyperparameters. A high extra variance signals a rapidly varying objective function that may lead to overtuning. They propose two stability-aware acquisition functions, Stable-UCB and Stable-EI, which penalize instability to encourage more robust HPCs.

Other works on early stopping in BO include Lorenz et al. (2016); Nguyen et al. (2017); Ishibashi et al. (2023); Li et al. (2023); Wilson (2024), although Ishibashi et al. (2023) is among the few that also directly considers overfitting in HPO. Their stopping criterion focuses on changes in the expected minimum simple regret, i.e., how much the estimated best objective improves with an additional function evaluation. As with Makarova et al. (2022, 2021), they measure outcomes using RYC and RTC but observe inconsistent results, indicating that while their method can cut computation time, it does not always prevent overtuning.

Fabris and Freitas (2019) conduct experiments with Auto-sklearn (Feurer et al., 2015) across 17 datasets, optimizing the area under the ROC curve. They distinguish among training, internal validation, and external test performance and frequently observe deteriorations from validation to test – phenomena they refer to as "meta-overfitting", especially when datasets are small (around 1000 or fewer observations). Although the validation–test correlation is generally high, the number of SMAC (Hutter et al., 2011) optimization iterations does not correlate with how severe this meta-overfitting is.

Earlier, Escalante et al. (2009) studied a particle swarm optimization (PSO)-based approach to full model selection, including preprocessing, feature selection, learner choice, and hyperparameter tuning. They note that while CV is the main safeguard against overfitting in their experiments, PSO's stochastic exploration can also mitigate the risk of pushing too hard on the validation error. Nonetheless, they acknowledge that repeated exploitation of CV estimates can cause validation improvements not always reflected on a held-out test set.

Lévesque (2018) undertook a large-scale support vector machine (SVM) HPO study with 118 datasets and identify overtuning as a serious limitation, especially in small-data scenarios. They test solutions like reshuffling, using an outer test set, and adopting posterior-mean-based selection in BO. Reshuffling helps in small-data regimes – particularly with holdout resampling – while choosing hyperparameters by posterior mean also yields better generalization. Selecting configurations on a separate selection set (Dos Santos et al., 2009; Koch et al., 2010; Igel, 2012), however, can hurt performance because it reduces the data available for HPO. Extending these findings, Nagler et al. (2024) provide a more rigorous analysis of reshuffling, demonstrating its benefits even for simple RS. They further analyze how reshuffling affects the validation loss landscape and derive regret bounds in the asymptotic regime.

Similarly, Larcher and Barbosa (2022) propose dynamic sampling holdout as a faster alternative to CV for AutoML. By reshuffling training and validation partitions at each generation, they reduce the variance and bias inherent in using the same splits repeatedly. Their empirical results show improvements in test performance and lower computational overhead.

Several foundational studies examine the estimation of GE and the variance of GE estimators. A thorough survey by Schulz-Kümpel et al. (2025) benchmarks a broad array of GE confidence-interval construction methods, while earlier and more recent contributions (Stone, 1974; Efron and Tibshirani, 1997; Bengio and Grandvalet, 2004; Austern and Zhou, 2020; Bayle et al., 2020; Bates et al., 2024; Paraschakis et al., 2024) provide theoretical and practical guidance on error estimation. A complementary survey on CV in model selection is offered by Arlot and Celisse (2010).

Empirical comparisons of resampling strategies include Molinaro et al. (2005), who find that in small-sample, high-dimensional genomic studies, naive resubstitution estimates are highly biased, but LOOCV, 10-fold CV, and the .632+ bootstrap can be more reliable. At the same time, the .632+ bootstrap may become biased if the signal-to-noise ratio is high. Further, Wainer and Cawley (2017) systematically evaluate 15 resampling-based HPO techniques for SVMs (with RBF kernels) and suggest that 2-fold or 3-fold CV is often a viable substitute for standard 5-fold CV, providing similar

generalization at reduced computational cost. In clinical prediction models, Dunias et al. (2024) show that standard 5-fold or 10-fold CV tends to yield robust out-of-sample discrimination and calibration, whereas the widely used 1SE rule (Breiman, 1984) can severely miscalibrate predictions in small or low-event-rate samples.

Blum and Hardt (2015) address overfitting to public leaderboards, where participants repeatedly adapt to holdout feedback. They propose the Ladder mechanism, which only reports improvements deemed statistically significant, reducing information leakage and thus mitigating overfitting. Extending this approach, Hardt (2017) introduce the Shaky Ladder, which adds randomized privacy guarantees so that participants cannot game small improvements. Neto et al. (2016) propose LadderBoot, which injects bootstrap noise to limit the sensitivity of public scores to repeated queries.

Another influential line of work in the context of overfitting in leaderboards leverages differential privacy. Dwork et al. (2015) present Thresholdout and SparseValidate, which provide theoretical generalization guarantees even after multiple adaptive queries to a holdout. Feldman et al. (2019) investigate how easily one can overfit a fixed test set in multiclass settings via adaptively chosen queries. While more classes raise the barrier to overfitting, it remains feasible with relatively few queries. In practice, Roelofs et al. (2019) analyze Kaggle competitions and, surprisingly, detect little evidence of large-scale overfitting, attributing poor generalization more to distribution shifts than to test set overuse.

Arora and Zhang (2021) explore this notion of "meta-overfitting" where continual reuse of a public benchmark – like ImageNet (Russakovsky et al., 2015) – gradually contaminates that benchmark. Researchers copy hyperparameters, architectures, or training procedures that appear to work well on the widely shared test set, thus implicitly optimizing on it. They propose an information-theoretic approach to quantify how much the test set is effectively "consumed" by repeated usage, suggesting that measuring a model's description length relative to a "pre-test-set" referee can help bound overfitting in such adaptive processes.

Quinlan and Cameron-Jones (1995) point out that more exhaustive searches during rule learning can degrade generalization – a phenomenon they term "oversearching". By fitting random idiosyncrasies in data, broader searches can lead to complex rules that fit the validation set but fail on new data. They propose a layered search strategy that expands search breadth incrementally and stops based on a probabilistic criterion, thereby avoiding the poor test performance often seen with exhaustive strategies.

Similarly, Reunanen (2003) shows that performing CV within variable selection can become self-defeating, because the repeated use of the same data splits to pick features leads to validation overfitting. Meanwhile, Loughrey and Cunningham (2005) note that aggressive search-based feature selection using methods like genetic algorithms can cause severe overfitting to the validation set, substantially harming test accuracy. They propose an early-stopping mechanism based on CV signals to limit the search depth before overfitting occurs.

Outside of the usage here, the phrase "meta-overfitting" often appears in meta-learning to indicate that knowledge acquired on source tasks may fail to generalize to new, target tasks (Yao et al., 2021; Hospedales et al., 2021; Huisman et al., 2021). It has also been discussed in the context of neural networks (Hospedales et al., 2021), AutoML systems (Yang et al., 2019, 2020), performance prediction (Loya et al., 2023), and other "learning to learn" settings (Barros et al., 2015; Chen et al., 2023; Song et al., 2024). This differs from the notion of meta-overfitting as used to describe consistent deterioration of test performance relative to validation performance within a single study or single dataset.

In the closely related field of algorithm configuration, Eggensperger et al. (2019) outline best practices and potential pitfalls, such as resource mismanagement, skewed evaluation metrics, or insufficient random seeds for robust performance assessment. They warn that insufficiently diverse evaluations – or evaluations that rely too heavily on a small set of training instances – risk

overtuning. They thus recommend strict train/test partitioning, averaging over multiple seeds, and using representative benchmarks.

Likewise, Eimer et al. (2023) point to limited reproducibility in reinforcement learning HPO. Optimizing hyperparameters on very few random seeds often causes severe overfitting, as configurations subsequently do poorly on unseen seeds. The authors advocate for adopting AutoML best practices, such as clear separation of tuning and evaluation seeds and employing systematic HPO strategies.

Many additional studies merely note overtuning or caution against it, especially in small, noisy data (as in certain linear or clinical models (Van Calster et al., 2020; Šinkovec et al., 2021; Riley et al., 2021)). While they do not directly measure overtuning, they nonetheless highlight the vulnerability of HPO to misleading improvements when sample sizes are too limited.

Finally, two survey works – Feurer and Hutter (2019) and Bischl et al. (2023) – explicitly identify overtuning as a core problem in HPO. They summarize various strategies for mitigating over-optimization of validation error, referencing much of the research above.

D Details on an Empirical Analysis of Overtuning

FCNet (Klein and Hutter, 2019) is based on exhaustive evaluations of fully connected feed-forward neural networks on four UCI regression datasets: Protein Structure, Slice Localization, Naval Propulsion, and Parkinsons Telemonitoring. Each dataset is randomly split into 60% training, 20% validation, and 20% test sets. The model architecture consists of two hidden layers followed by a linear output layer. For the search space and additional information, see Klein and Hutter (2019). Each configuration (a combination of architectural and training hyperparameters) is trained using the Adam optimizer for 100 epochs, minimizing the mean squared error (MSE), which is also used as the evaluation metric. To account for stochasticity in training, each configuration is repeated four times using different random seeds. This yields a tabular benchmark dataset with complete learning curves and performance statistics for all configurations. We use the final (with respect to the number of epochs trained) validation and test MSE in our analyses. For each replication and dataset combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T = 62208$).

LCBench (Zimmer et al., 2021) is based on evaluating 2000 HPCs, sampled uniformly at random, for a funnel-shaped MLP on 35 classification datasets. For the search space and additional information, see Zimmer et al. (2021). Each dataset reserves 33% as a test set, and the remaining data is split into training and validation sets, with the validation set comprising 33%. Models are trained using SGD with cosine annealing (without restarts) and evaluated using accuracy and cross-entropy. We use the final (with respect to the number of epochs trained) validation and test performance values in our analyses. For each dataset and metric combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T = 2000$).

WDTB (Grinsztajn et al., 2022) includes different learning algorithms evaluated on a curated benchmark of 45 datasets, categorized into four groups: categorical classification, numerical classification, categorical regression, and numerical regression, where categorical/numerical refers to the feature types. Learning algorithms include Random Forest, XGBoost, Gradient Boosting Tree, ResNet, FT Transformer, SAINT, MLP, and HistGradientBoostingTree. For the search spaces and additional information, see Grinsztajn et al. (2022). For each learning algorithm, RS with approximately 400 HPC evaluations is performed, beginning with a default HPC. The data splitting and evaluation protocol is designed to ensure fair and efficient comparison across datasets: 70% of samples are allocated to the training set (unless this exceeds a predefined maximum), and the remaining 30% is split into 30% validation and 70% test sets, both capped at 50000 samples. The validation set is used exclusively for selecting the best configuration during RS and is distinct from the internal validation set used for early stopping. To adjust for dataset size variability, the number of evaluation folds depends on the number of test samples: one fold for $>$6000 samples, two for 3000–6000, three for 1000–3000, and five for $<$1000. All models are evaluated on the same folds to ensure comparability. Performance is measured using accuracy (for classification) and R^2 (for regression). In our analyses, we exclude default HPCs and use only the random HPCs. For each learning algorithm and dataset combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T \approx 400$).

TabZilla (McElfresh et al., 2023) includes evaluations of various learning algorithms on a total of 176 classification datasets. Learning algorithms include CatBoost, XGBoost, LightGBM, DeepFM, DANet, FT Transformer, TabTransformer, two MLP variants, NODE, ResNet, SAINT, STG, TabNet, TabPFN (no HPO), VIME, NAM, Decision Tree, KNN, Logistic Regression (Linear Model, no HPO), Random Forest, and SVM. For the search spaces and additional information, see McElfresh et al. (2023). Each dataset uses the 10 train/test folds provided by OpenML. Within each training fold, a further split is used to construct a validation set for HPO. The best configuration is selected based on validation performance, and final performance is reported on the test set without retraining. Models are evaluated using accuracy, F1, log loss, and ROC AUC. In our analyses, we exclude runs

with fewer than 30 HPC evaluations and exclude the default HPCs, retaining only the random HPCs. For each learning algorithm, dataset, fold, and metric combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T \approx 29$).

TabRepo (Salinas and Erickson, 2024) includes evaluations of 1530 learning algorithm and HPC combinations across 211 classification and regression datasets. We use the "D244_F3_C1530_175" context, restricted to 175 datasets. Learning algorithms include Random Forest, Extra Trees, LightGBM, XGBoost, CatBoost, Linear Model, KNN, and two neural architectures. For the search spaces and additional information, see Salinas and Erickson (2024). Models are evaluated using 3-fold CV. For each fold, data is split into 90% training and 10% test. All models are trained with bagging, generating out-of-fold predictions for estimating generalization performance. Performance is measured using ROC AUC (for binary classification), log loss (for multi-class classification), and RMSE (for regression). Each algorithm has one default HPC and 200 configurations sampled uniformly at random. In our analyses, we exclude runs with fewer than 30 HPCs. For each learning algorithm, dataset, and fold combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T \geq 30$).

reshuffling (Nagler et al., 2024) evaluates four learning algorithms (Elastic Net, Funnel MLP, XGBoost, CatBoost) on ten binary classification tasks, varying the dataset size, resampling strategy, reshuffling status, and optimizer. For the search spaces and additional information, see Nagler et al. (2024). A fixed outer test set of 5000 samples is held out and never used during HPO. For HPO, subsets of the remaining data are drawn with training-validation sizes $n \in \{500, 1000, 5000\}$. Resampling strategies include: 80/20 holdout, 5-fold CV, 5x 80/20 holdout, and 5x 5-fold CV, ensuring constant train/validation sizes but varying the splits. Performance is measured via accuracy, log loss, and ROC AUC (BO uses ROC AUC only). The best configuration is retrained on the full HPO data and evaluated on the outer test set. RS is performed with 500 fixed HPCs per replication (10 total). In our analyses, we use only the RS runs without reshuffled resampling. We revisit BO (HEBO and SMAC for a budget of 250 HPCs) and reshuffling the resampling splits in Section 6. For each learning algorithm, dataset, dataset size, repetition, resampling, and metric combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T = 500$).

PD1 (Wang et al., 2024) is a large-scale HPO dataset developed for evaluating BO algorithms in deep learning. It consists of 24 tasks, each defined by a dataset (e.g., CIFAR10, ImageNet), a model (e.g., ResNet50, Transformer), and a batch size (determined by hardware). For each task, approximately 500 "matched" and 1500 "unmatched" HPCs are evaluated from a shared four-dimensional search space: learning rate (log scale), momentum (log scale), polynomial decay power, and decay fraction. All tasks use Nesterov momentum with fixed pipelines, varying only optimizer hyperparameters. Each configuration is fully trained and logged with learning curves, including validation cross-entropy loss, error rate, and divergence status. Performance metrics are given by error rate and cross-entropy. We use the "phase1" data (both "matched" and "unmatched"). We exclude ImageNet ResNet50 (all batch sizes), LM1B Transformer (2048), WMT15 German-English xformer (64), and UniRef50 Transformer (128), leaving 18 tasks due to failed/incomplete runs or insufficient full-epoch HPCs or runs where test performance was not available. We use the final validation and test performances for each task in our analyses. For each task and metric combination, we compute the relative overtuning as defined in Definition 3.1 based on an HPC trajectory of all evaluated HPCs ($T \geq 1300$).

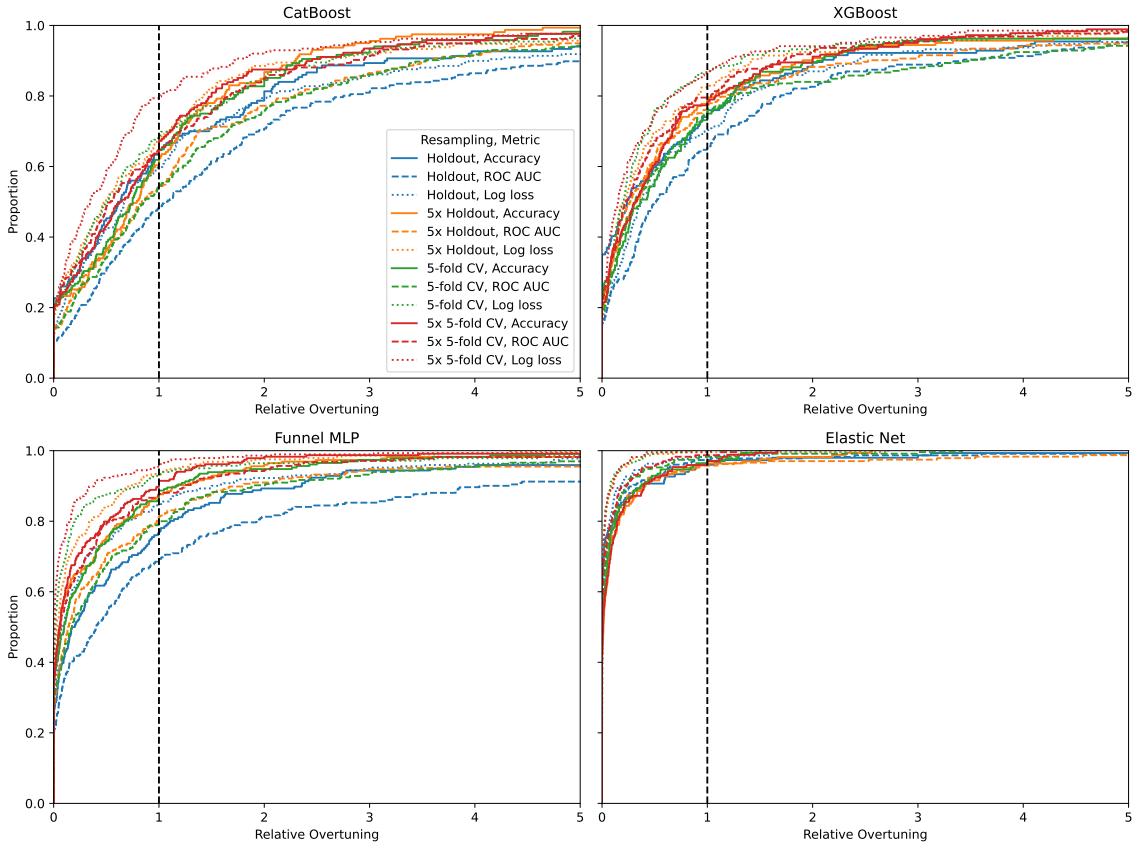


Figure 3: ECDFs of relative overtuning for *reshuffling* (Nagler et al., 2024). Stratified for the learning algorithm, resampling method and performance metric but not dataset sizes.

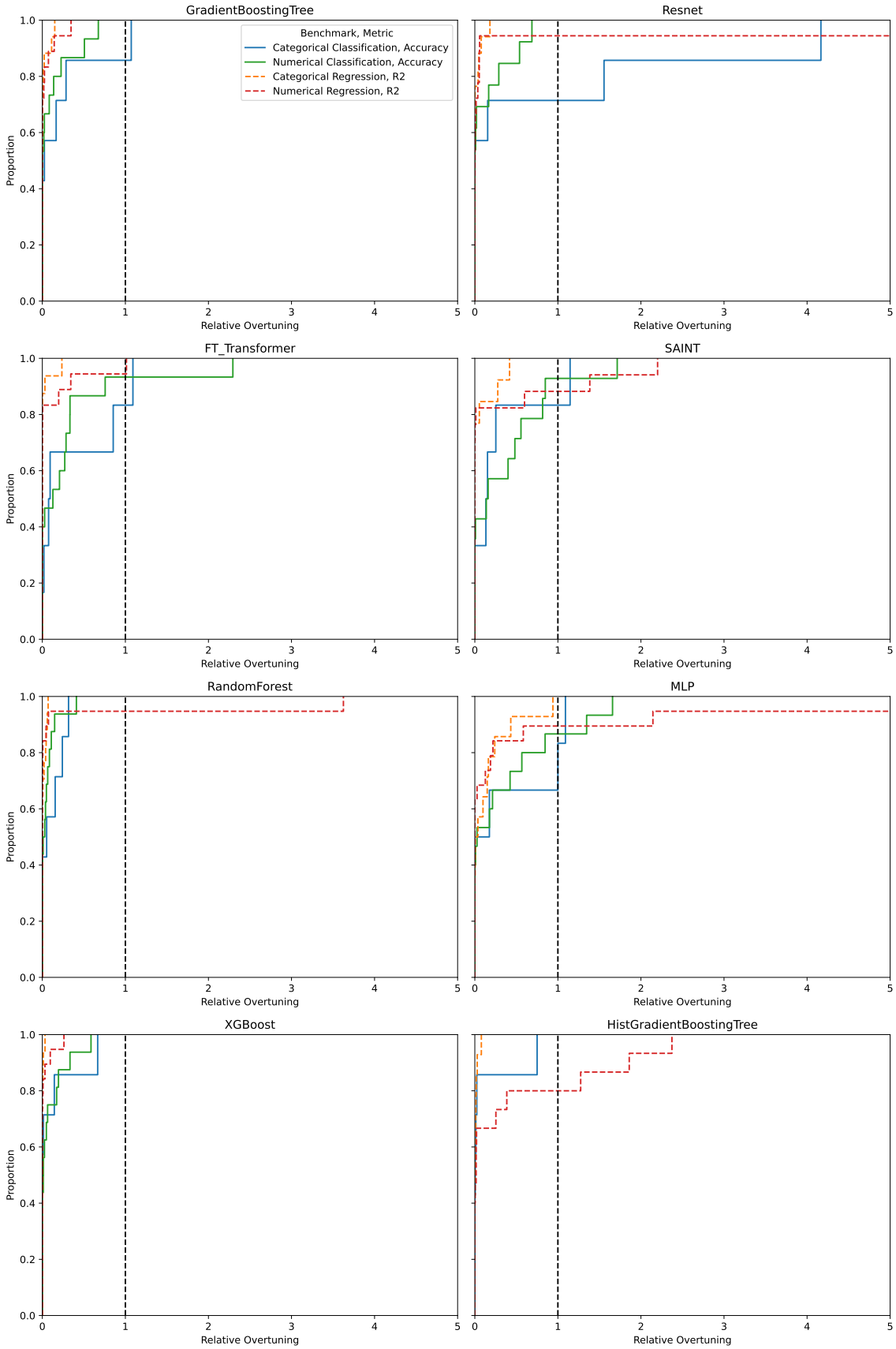


Figure 4: ECDFs of relative overtuning for *WDTB* (Grinsztajn et al., 2022). Stratified for the learning algorithm, benchmark type and performance metric.

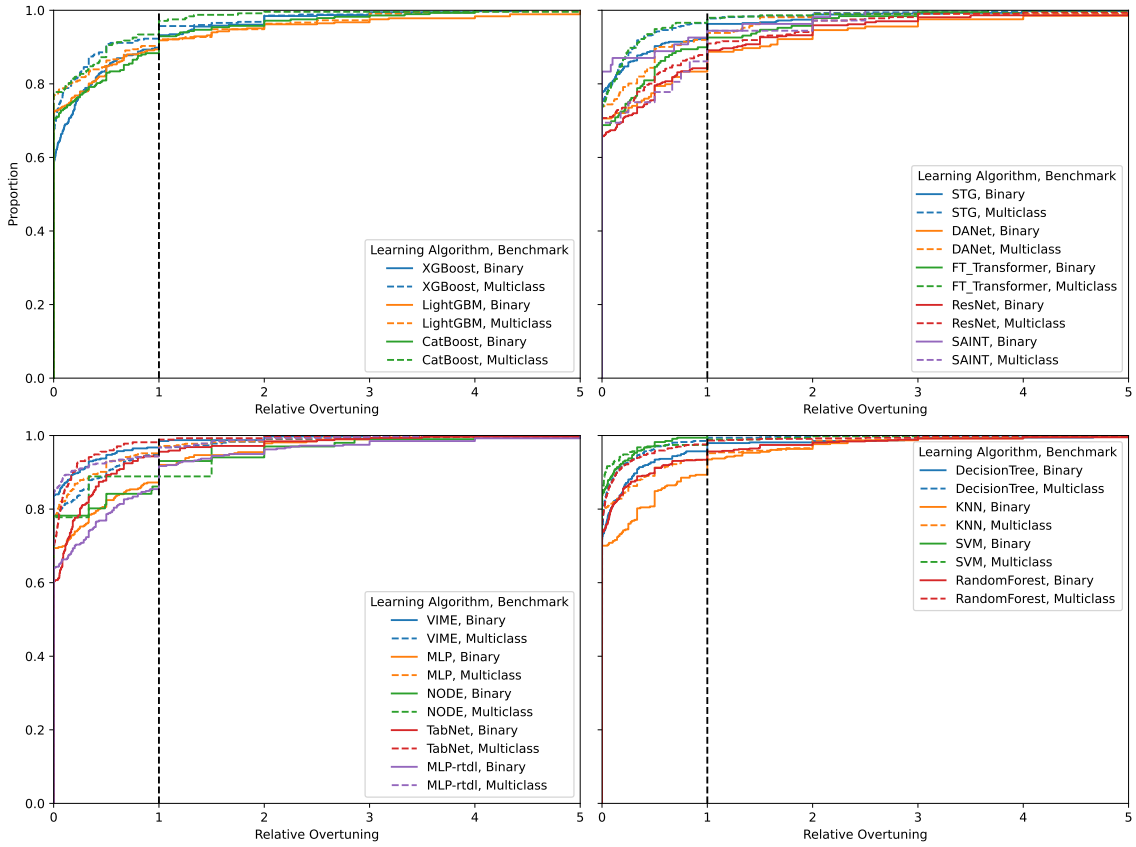


Figure 5: ECDFs of relative overtuning for *TabZilla* (McElfresh et al., 2023). Performance metric accuracy. Stratified for the learning algorithm, and benchmark type.

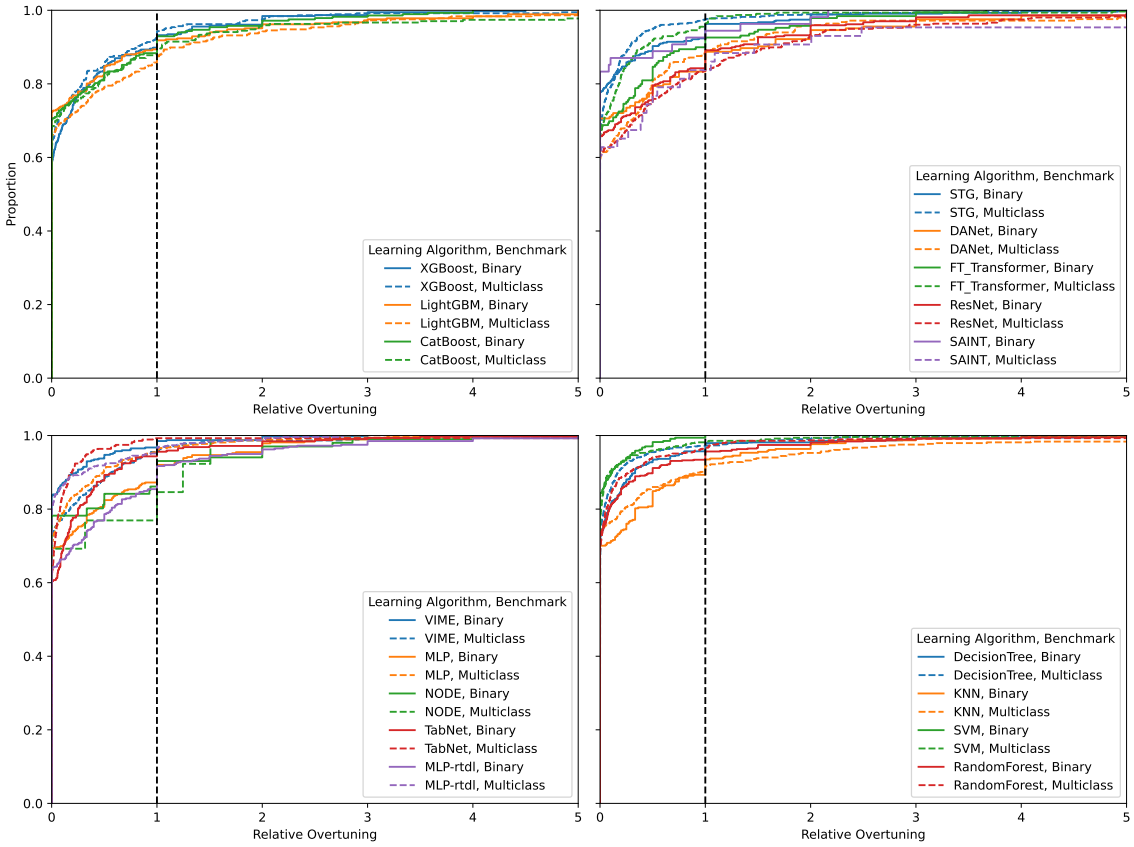


Figure 6: ECDFs of relative overtuning for *TabZilla* (McElfresh et al., 2023). Performance metric F1. Stratified for the learning algorithm, and benchmark type.

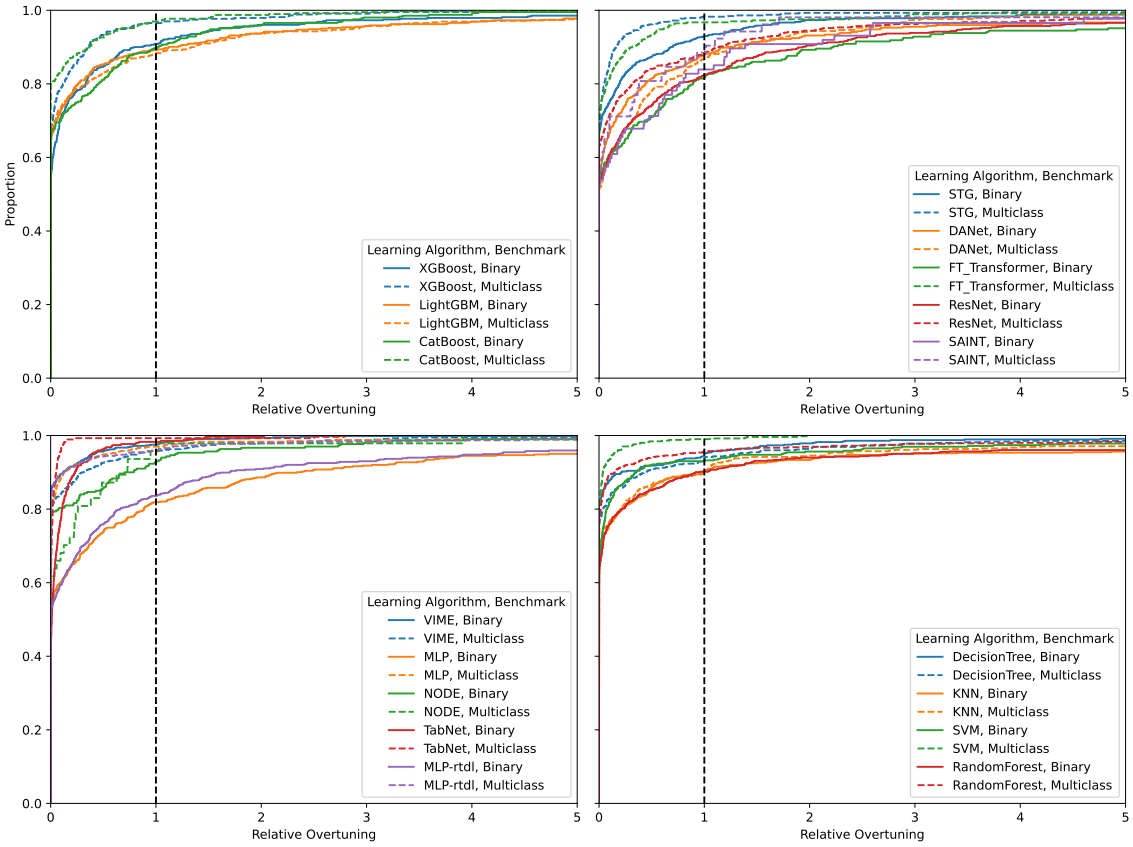


Figure 7: ECDFs of relative overtuning for *TabZilla* (McElfresh et al., 2023). Performance metric log loss. Stratified for the learning algorithm, and benchmark type.

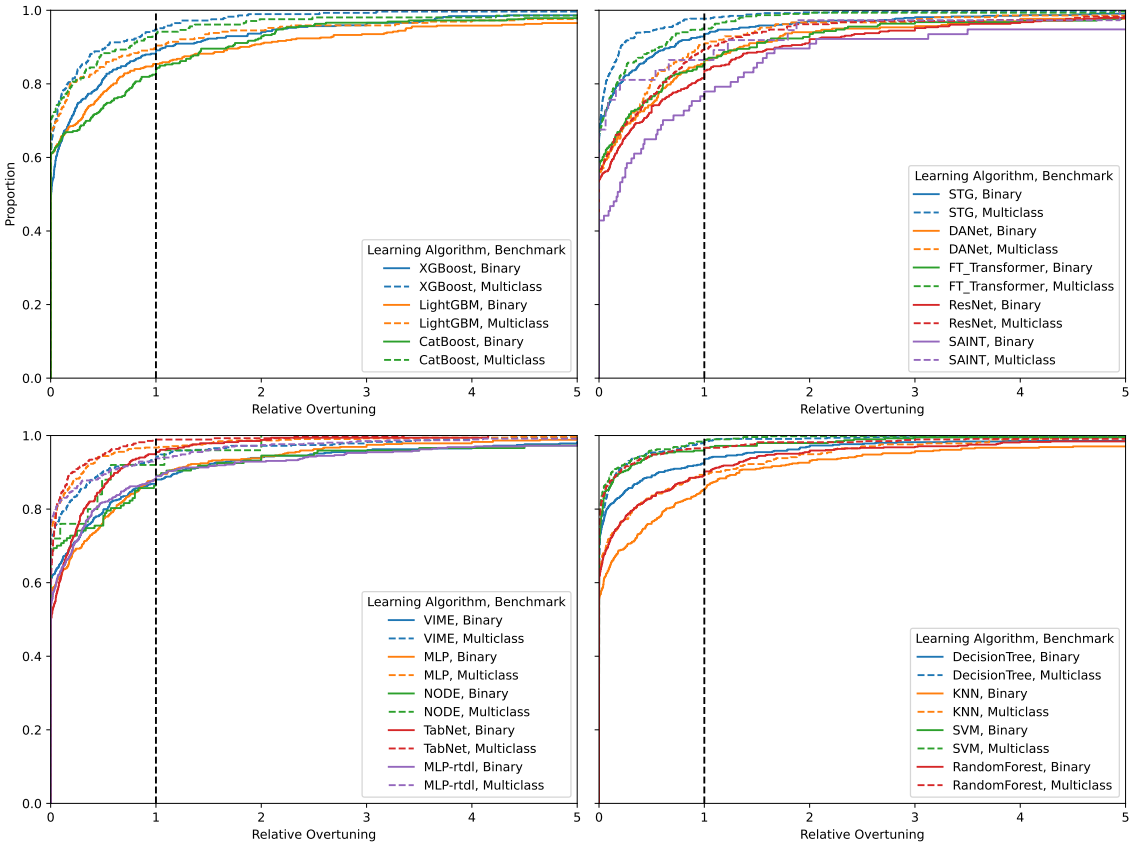


Figure 8: ECDFs of relative overtuning for *TabZilla* (McElfresh et al., 2023). Performance metric ROC AUC. Stratified for the learning algorithm, and benchmark type.

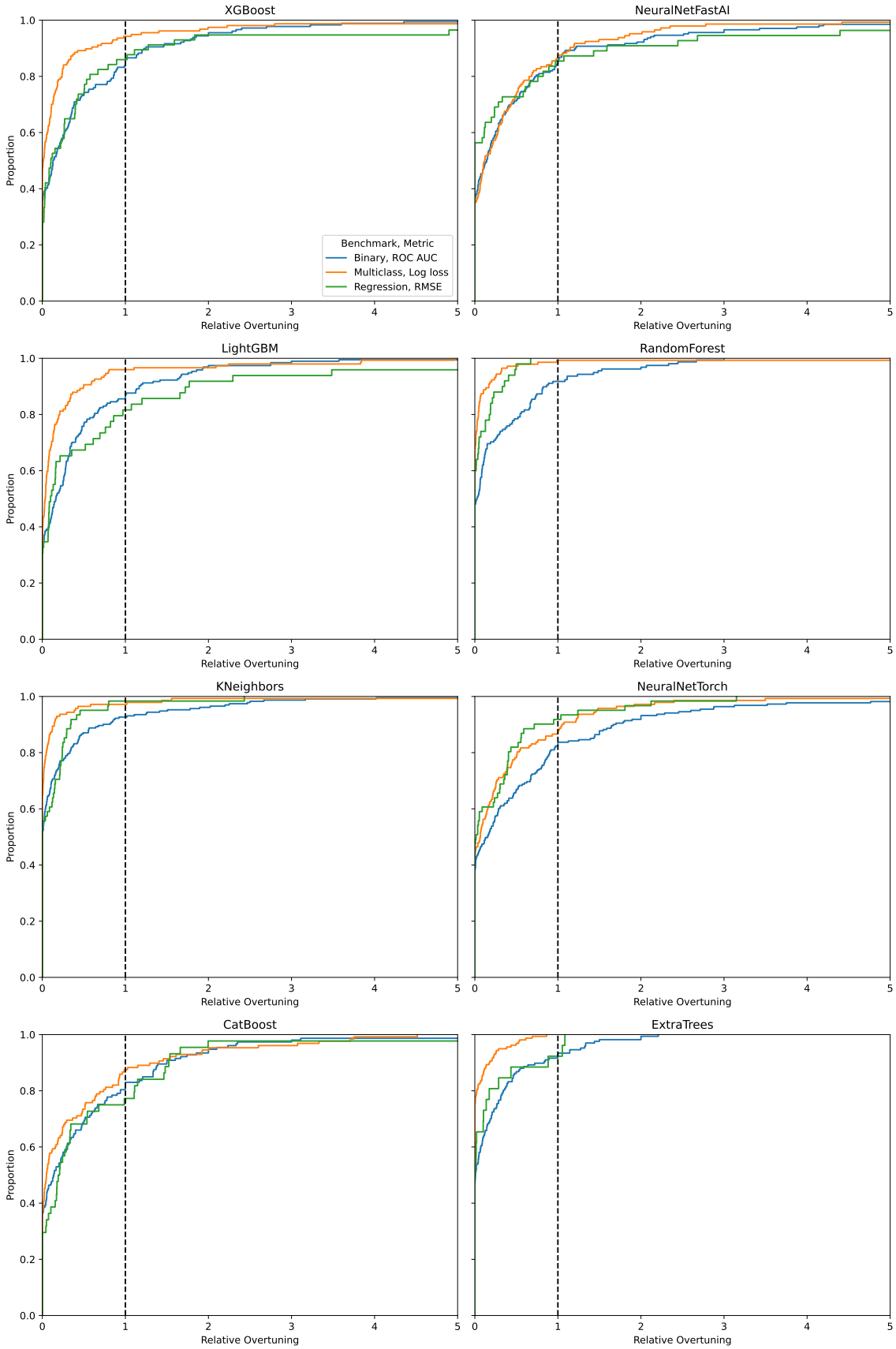


Figure 9: ECDFs of relative overtuning for *TabRepo* (Salinas and Erickson, 2024). Stratified for the learning algorithm, benchmark type and performance metric.

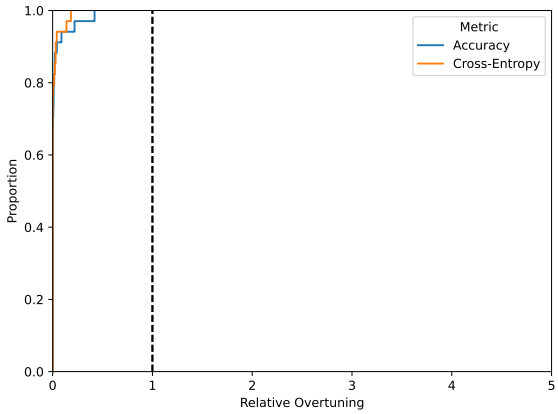


Figure 10: ECDFs of relative overtuning for *LCBench* (Zimmer et al., 2021). Stratified for the performance metric.

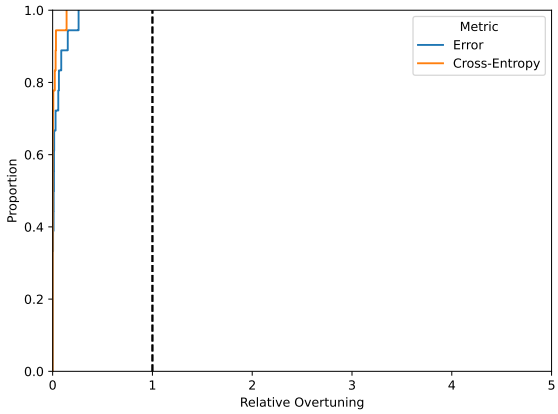


Figure 11: ECDFs of relative overtuning for *PD1* (Wang et al., 2024). Stratified for the performance metric.

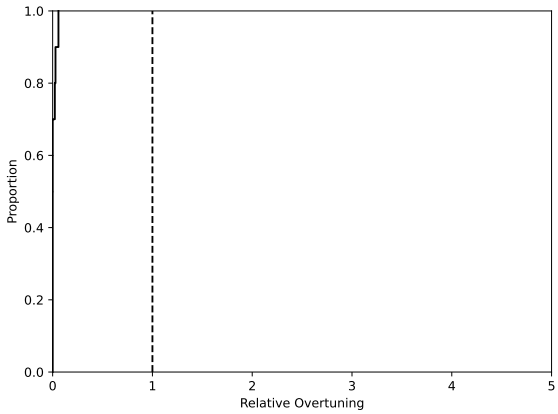


Figure 12: ECDFs of relative overtuning for *FCNet* (Klein and Hutter, 2019).

E Details on Modeling the Determinants of Overtuning

For an introduction to general linear mixed-effects models, we refer the reader to McCulloch et al. (2008) and Bates et al. (2015). All statistical analyses are interpreted at a significance level of $\alpha = 0.05$ However, we emphasize that we perform many analyses and many of these analyses are conducted on large datasets. As such, statistical significance should be interpreted with caution, as even negligible effects may appear significant due to the large sample sizes. Nonetheless, the magnitude of coefficients as well as associated z- and t-statistics can still provide meaningful insights into potentially relevant determinants. Finally, we stress that our analysis is exploratory in nature and does not involve the confirmation of pre-specified hypotheses (Herrmann et al., 2024).

(a) Fixed effects results from a GLMM predicting probability of nonzero overtuning.

Predictor	Estimate	Std. Error	z value	p-value
(intercept)	-1.161680	0.170379	-6.818	< 0.001
budget	3.073490	0.054291	56.612	< 0.001
budget^2	-2.034155	0.050771	-40.065	< 0.001
classifier (CatBoost)	1.939010	0.011118	174.395	< 0.001
classifier (Funnel MLP)	1.458479	0.010659	136.827	< 0.001
classifier (XGBoost)	1.606069	0.010774	149.066	< 0.001
resampling (5x holdout)	-0.300831	0.010779	-27.908	< 0.001
resampling (5-fold CV)	-0.374434	0.010762	-34.793	< 0.001
resampling (5x 5-fold CV)	-0.467436	0.010747	-43.493	< 0.001
dataset size (1000)	-0.290248	0.009386	-30.924	< 0.001
dataset size (5000)	-0.981109	0.009358	-104.839	< 0.001
optimizer (HEBO)	0.083319	0.009208	9.049	< 0.001
optimizer (SMAC)	0.188119	0.009245	20.347	< 0.001

(b) Fixed effects results from an LMM predicting nonzero relative overtuning on log scale.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	-1.757e+00	1.654e-01	1.534e+01	-10.622	< 0.001
budget	3.274e-01	5.351e-02	1.542e+05	6.119	< 0.001
budget^2	-1.681e-01	4.782e-02	1.542e+05	-3.515	< 0.001
classifier (CatBoost)	2.151e+00	1.136e-02	1.542e+05	189.408	< 0.001
classifier (Funnel MLP)	1.061e+00	1.125e-02	1.542e+05	94.266	< 0.001
classifier (XGBoost)	1.554e+00	1.143e-02	1.542e+05	135.941	< 0.001
resampling (5x Holdout)	-3.350e-01	9.392e-03	1.542e+05	-35.666	< 0.001
resampling (5-fold CV)	-3.544e-01	9.428e-03	1.542e+05	-37.594	< 0.001
resampling (5x 5-fold CV)	-4.969e-01	9.478e-03	1.542e+05	-52.423	< 0.001
dataset size (1000)	-1.973e-01	7.924e-03	1.542e+05	-24.905	< 0.001
dataset size (5000)	-5.188e-01	8.519e-03	1.542e+05	-60.901	< 0.001
optimizer (HEBO)	-3.011e-01	8.281e-03	1.542e+05	-36.363	< 0.001
optimizer (SMAC)	-2.581e-01	8.336e-03	1.542e+05	-30.956	< 0.001

Table 3: Fixed effects results of mixed models used to analyze overtuning. BO and RS runs, no reshuffling, test performance of the model retrained on all data. Reference levels: Elastic Net (classifier), holdout (resampling), 500 (dataset size), RS (optimizer).

(a) Fixed effects results from an LMM predicting final meta-overfitting.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	5.915e-02	1.083e-02	9.387e+00	5.462	< 0.001
classifier (CatBoost)	3.767e-02	1.049e-03	1.437e+04	35.892	< 0.001
classifier (Funnel MLP)	2.839e-02	1.049e-03	1.437e+04	27.052	< 0.001
classifier (XGBoost)	1.908e-02	1.049e-03	1.437e+04	18.182	< 0.001
resampling (5x Holdout)	-2.929e-02	1.049e-03	1.437e+04	-27.905	< 0.001
resampling (5-fold CV)	-3.115e-02	1.049e-03	1.437e+04	-29.685	< 0.001
resampling (5x 5-fold CV)	-4.445e-02	1.049e-03	1.437e+04	-42.355	< 0.001
dataset size (1000)	-2.122e-02	9.089e-04	1.437e+04	-23.351	< 0.001
dataset size (1000)	-4.519e-02	9.089e-04	1.437e+04	-49.718	< 0.001
optimizer (HEBO)	1.623e-03	9.089e-04	1.437e+04	1.786	0.074
optimizer (SMAC)	3.793e-03	9.089e-04	1.437e+04	4.173	< 0.001

(b) Fixed effects results from an LMM predicting final test regret.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	2.215e-02	4.050e-03	9.417e+00	5.468	< 0.001
classifier (CatBoost)	1.048e-02	4.633e-04	1.437e+04	22.624	< 0.001
classifier (Funnel MLP)	1.739e-02	4.633e-04	1.437e+04	37.544	< 0.001
classifier (XGBoost)	5.410e-03	4.633e-04	1.437e+04	11.679	< 0.001
resampling (5x Holdout)	-6.310e-03	4.633e-04	1.437e+04	-13.621	< 0.001
resampling (5-fold CV)	-7.283e-03	4.633e-04	1.437e+04	-15.722	< 0.001
resampling (5x 5-fold CV)	-8.857e-03	4.633e-04	1.437e+04	-19.118	< 0.001
dataset size (1000)	-6.841e-03	4.012e-04	1.437e+04	-17.053	< 0.001
dataset size (5000)	-1.636e-02	4.012e-04	1.437e+04	-40.782	< 0.001
optimizer (HEBO)	-2.073e-03	4.012e-04	1.437e+04	-5.167	< 0.001
optimizer (SMAC)	-2.773e-04	4.012e-04	1.437e+04	-0.691	0.489

Table 4: Fixed effects results of mixed models used to analyze final meta-overfitting and test regret. BO and RS runs, no reshuffling, test performance of the model retrained on all data. Reference levels of factors are: Elastic Net (classifier), holdout (resampling), 500 (dataset size), RS (optimizer).

(a) Fixed effects results from a GLMM predicting probability of nonzero overtuning.

Predictor	Estimate	Std. Error	z value	p-value
(intercept)	-0.368810	0.206555	-1.786	0.074
classifier (CatBoost)	1.740219	0.133810	13.005	< 0.001
classifier (Funnel MLP)	1.813526	0.134520	13.481	< 0.001
classifier (XGBoost)	1.716060	0.133593	12.845	< 0.001
dataset size (1000)	-0.376622	0.113395	-3.321	< 0.001
dataset size (5000)	-1.066268	0.114042	-9.350	< 0.001
optimizer (HEBO + ES)	-0.549139	0.092045	-5.966	< 0.001

(b) Fixed effects results from an LMM predicting nonzero relative overtuning on log scale.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	-1.866e+00	2.047e-01	3.707e+01	-9.117	< 0.001
classifier (CatBoost)	1.794e+00	1.493e-01	9.905e+02	12.017	< 0.001
classifier (Funnel MLP)	6.331e-01	1.427e-01	9.859e+02	4.436	< 0.001
classifier (XGBoost)	1.197e+00	1.447e-01	9.872e+02	8.275	< 0.001
dataset size (1000)	-1.760e-01	9.711e-02	9.823e+02	-1.812	0.070
dataset size (5000)	-4.765e-01	1.065e-01	9.870e+02	-4.475	< 0.001
optimizer (HEBO + ES)	-2.753e-01	8.414e-02	9.806e+02	-3.272	0.001

Table 5: Fixed effects results of mixed models used to analyze overtuning. BO runs (only HEBO and HEBO with early stopping on 5-fold CV and ROC AUC as performance metric), no reshuffling, test performance of the model retrained on all data. Reference levels: Elastic Net (classifier), 500 (dataset size), HEBO (optimizer). Analyses performed for the final time point which may differ between HEBO and HEBO with early stopping.

(a) Fixed effects results from a GLMM predicting probability of nonzero overtuning.

Predictor	Estimate	Std. Error	z value	p-value
(intercept)	-1.271737	0.082493	-15.416	< 0.001
budget	2.239491	0.025025	89.489	< 0.001
budget^2	-1.481421	0.023734	-62.418	< 0.001
metric (ROC AUC)	0.650435	0.004382	148.433	< 0.001
metric (log loss)	0.295035	0.004336	68.050	< 0.001
classifier (CatBoost)	1.390966	0.005180	268.533	< 0.001
classifier (Funnel MLP)	1.111329	0.005116	217.220	< 0.001
classifier (XGBoost)	1.183944	0.005128	230.857	< 0.001
resampling (5x Holdout)	-0.256786	0.005042	-50.934	< 0.001
resampling (5-fold CV)	-0.302036	0.005041	-59.920	< 0.001
resampling (5x 5-fold CV)	-0.491480	0.005048	-97.353	< 0.001
dataset size (500)	-0.256418	0.004357	-58.845	< 0.001
dataset size (1000)	-0.682647	0.004381	-155.814	< 0.001
reshuffled (TRUE)	0.043902	0.003555	12.351	< 0.001

(b) Fixed effects results from an LMM predicting nonzero relative overtuning on log scale.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	-1.563e+00	1.375e-01	1.689e+01	-11.367	< 0.001
budget	2.604e-01	2.952e-02	5.379e+05	8.819	< 0.001
budget^2	-1.293e-01	2.695e-02	5.379e+05	-4.796	< 0.001
metric (ROC AUC)	-2.200e-01	4.978e-03	5.379e+05	-44.189	< 0.001
metric (log loss)	-6.831e-01	5.118e-03	5.379e+05	-133.462	< 0.001
classifier (CatBoost)	2.118e+00	6.156e-03	5.379e+05	344.120	< 0.001
classifier (Funnel MLP)	7.089e-01	6.082e-03	5.379e+05	116.555	< 0.001
classifier (XGBoost)	1.691e+00	6.355e-03	5.379e+05	266.063	< 0.001
resampling (5x Holdout)	-2.494e-01	5.305e-03	5.379e+05	-47.000	< 0.001
resampling (5-fold CV)	-2.614e-01	5.319e-03	5.379e+05	-49.135	< 0.001
resampling (5x 5-fold CV)	-4.582e-01	5.440e-03	5.379e+05	-84.232	< 0.001
dataset size (500)	-1.048e-01	4.518e-03	5.379e+05	-23.193	< 0.001
dataset size (1000)	-3.331e-01	4.803e-03	5.379e+05	-69.363	< 0.001
reshuffled (TRUE)	5.150e-02	3.827e-03	5.379e+05	13.458	< 0.001

Table 6: Fixed effects results of mixed models used to analyze overtuning. RS runs, test performance of the model retrained on all data. Reference levels: accuracy (metric) Elastic Net (classifier), holdout (resampling), 500 (dataset size), FALSE (reshuffled).

(a) Fixed effects results from an LMM predicting final meta-overfitting.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	4.463e-02	3.665e-03	1.048e+01	12.178	< 0.001
metric (ROC AUC)	2.973e-02	5.864e-04	2.877e+04	50.697	< 0.001
metric (log loss)	1.699e-04	5.864e-04	2.877e+04	0.290	0.772
classifier (CatBoost)	1.398e-02	6.771e-04	2.877e+04	20.648	< 0.001
classifier (Funnel MLP)	1.051e-02	6.771e-04	2.877e+04	15.517	< 0.001
classifier (XGBoost)	9.060e-03	6.771e-04	2.877e+04	13.381	< 0.001
resampling (5x Holdout)	-2.839e-02	6.771e-04	2.877e+04	-41.928	< 0.001
resampling (5-fold CV)	-3.596e-02	6.771e-04	2.877e+04	-53.119	< 0.001
resampling (5x 5-fold CV	-4.633e-02	6.771e-04	2.877e+04	-68.421	< 0.001
dataset size (500)	-1.540e-02	5.864e-04	2.877e+04	-26.264	< 0.001
dataset size (1000)	-3.287e-02	5.864e-04	2.877e+04	-56.050	< 0.001
reshuffled (TRUE)	1.672e-02	4.788e-04	2.877e+04	34.916	< 0.001

(b) Fixed effects results from an LMM predicting final test regret.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	1.119e-02	1.362e-03	1.066e+01	8.217	< 0.001
metric (ROC AUC)	1.095e-02	2.742e-04	2.877e+04	39.950	< 0.001
metric (log loss)	1.022e-03	2.742e-04	2.877e+04	3.726	< 0.001
classifier (CatBoost)	5.123e-03	3.166e-04	2.877e+04	16.181	< 0.001
classifier (Funnel MLP)	1.058e-02	3.166e-04	2.877e+04	33.418	< 0.001
classifier (XGBoost)	3.237e-03	3.166e-04	2.877e+04	10.225	< 0.001
resampling (5x Holdout)	-4.994e-03	3.166e-04	2.877e+04	-15.772	< 0.001
resampling (5-fold CV)	-5.131e-03	3.166e-04	2.877e+04	-16.205	< 0.001
resampling (5x 5-fold CV	-6.882e-03	3.166e-04	2.877e+04	-21.736	< 0.001
dataset size (500)	-5.088e-03	2.742e-04	2.877e+04	-18.554	< 0.001
dataset size (500)	-1.030e-02	2.742e-04	2.877e+04	-37.571	< 0.001
reshuffled (TRUE)	-1.529e-04	2.239e-04	2.877e+04	-0.683	0.495

Table 7: Fixed effects results of mixed models used to analyze final meta-overfitting and test regret. RS runs, test performance of the model retrained on all data. Reference levels: accuracy (metric) Elastic Net (classifier), holdout (resampling), 500 (dataset size), FALSE (reshuffled).

(a) Fixed effects results from a GLMM predicting probability of nonzero overtuning.

Predictor	Estimate	Std. Error	z value	p-value
(intercept)	-0.903880	0.153342	-5.895	< 0.001
budget	2.495057	0.092010	27.117	< 0.001
budget^2	-1.626134	0.088101	-18.458	< 0.001
classifier (CatBoost)	1.799761	0.018983	94.808	< 0.001
classifier (Funnel MLP)	1.426324	0.018159	78.545	< 0.001
classifier (XGBoost)	1.552415	0.018403	84.356	< 0.001
dataset size (500)	-0.049298	0.016481	-2.991	0.003
dataset size (1000)	-0.664234	0.016094	-41.272	< 0.001
reshuffled (TRUE)	-0.264457	0.013187	-20.054	< 0.001

(b) Fixed effects results from an LMM predicting nonzero relative overtuning on log scale.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	-1.884e+00	1.472e-01	1.758e+01	-12.792	< 0.001
budget	2.229e-01	8.841e-02	5.548e+04	2.521	0.012
budget^2	-1.417e-01	8.109e-02	5.548e+04	-1.748	0.081
classifier (CatBoost)	2.080e+00	1.842e-02	5.549e+04	112.902	< 0.001
classifier (Funnel MLP)	1.492e+00	1.861e-02	5.549e+04	80.164	< 0.001
classifier (XGBoost)	1.710e+00	1.920e-02	5.549e+04	89.042	< 0.001
dataset size (500)	-6.258e-02	1.376e-02	5.548e+04	-4.549	< 0.001
dataset size (1000)	-3.982e-01	1.462e-02	5.548e+04	-27.232	< 0.001
reshuffled (TRUE)	-2.693e-01	1.159e-02	5.548e+04	-23.236	< 0.001

Table 8: Fixed effects results of mixed models used to analyze overtuning. RS runs, subset of runs with holdout and ROC AUC as performance metric, test performance of the model retrained on all data. Reference levels: Elastic Net (classifier), 500 (dataset size), FALSE (reshuffled).

(a) Fixed effects results from an LMM predicting final meta-overfitting.

Predictor	Estimate	Std. Error	df	t value	p-value
(Intercept)	8.699e-02	2.010e-02	9.338e+00	4.329	0.002
classifier (CatBoost)	3.000e-02	3.055e-03	2.375e+03	9.820	< 0.001
classifier (Funnel MLP)	4.230e-02	3.055e-03	2.375e+03	13.845	< 0.001
classifier (XGBoost)	2.066e-02	3.055e-03	2.375e+03	6.761	< 0.001
dataset size (500)	-4.254e-02	2.646e-03	2.375e+03	-16.077	< 0.001
dataset size (1000)	-9.853e-02	2.646e-03	2.375e+03	-37.241	< 0.001
reshuffled (TRUE)	5.483e-02	2.160e-03	2.375e+03	25.382	< 0.001

(b) Fixed effects results from an LMM predicting final test regret.

Predictor	Estimate	Std. Error	df	t value	p-value
(intercept)	2.330e-02	4.770e-03	1.022e+01	4.885	< 0.001
classifier (CatBoost)	9.145e-03	1.335e-03	2.375e+03	6.849	< 0.001
classifier (Funnel MLP)	2.310e-02	1.335e-03	2.375e+03	17.303	< 0.001
classifier (XGBoost)	7.782e-03	1.335e-03	2.375e+03	5.828	< 0.001
dataset size (500)	-7.634e-03	1.156e-03	2.375e+03	-6.602	< 0.001
dataset size (1000)	-1.915e-02	1.156e-03	2.375e+03	-16.562	< 0.001
reshuffled (TRUE)	-5.656e-03	9.442e-04	2.375e+03	-5.990	< 0.001

Table 9: Fixed effects results of mixed models used to analyze final meta-overfitting and test regret. RS runs, subset of runs with holdout and ROC AUC as performance metric, test performance of the model retrained on all data. Reference levels: Elastic Net (classifier), 500 (dataset size), FALSE (reshuffled).

E.1 HEBO vs. HEBO with Early Stopping

As mentioned in Appendix A, when HPO protocols follow different search trajectories – due to factors like early stopping, choice of optimizer, or resource constraints – it is necessary to compare the test performance of their incumbents to assess generalization properly since overtuning cannot capture this performance aspect. In Section 6 we have seen that HEBO with early stopping à la Makarova et al. (2022) reduces overtuning in the *reshuffling* study (Nagler et al., 2024) based on 5-fold CV and ROC AUC as performance metric (other factors left at their default, i.e., non-reshuffled resampling and test performance assessed via retraining the inducer configured by a given HPC on all data and evaluating on the outer holdout set). We also visualize this (over all learning algorithms but stratified) for the dataset size in Figure 13 where we observe that HEBO with early stopping indeed exhibits less overtuning.

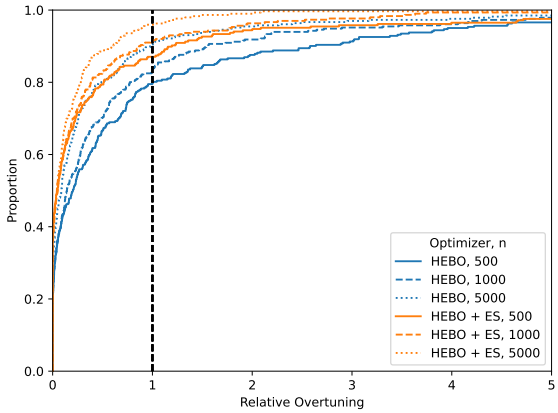
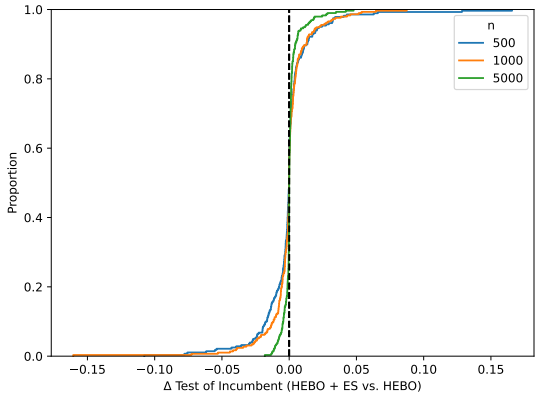


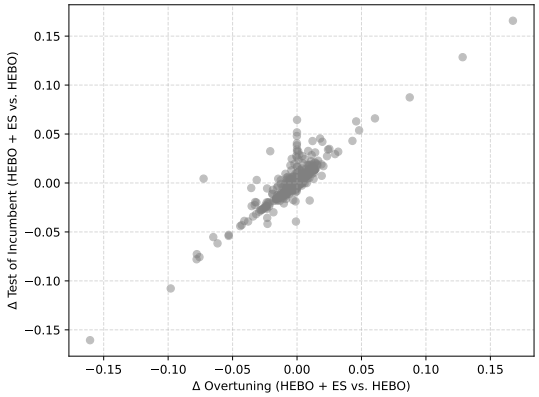
Figure 13: ECDF of relative overtuning for HEBO vs. HEBO with early stopping based on the *reshuffling* study (Nagler et al., 2024). 5-fold CV as resampling. ROC AUC as performance metric.

However, looking at the difference in test performance of the final incumbent returned by HEBO vs. HEBO with early stopping (Figure 14), we observe that HEBO with early stopping does not consistently improve generalization performance. As shown in Figure 14a, HEBO with early stopping yields worse test performance (positive Δ) nearly as often as it yields better test performance (negative Δ) compared to HEBO without early stopping.

To further understand the impact of early stopping on generalization, we analyze the relationship between changes in overtuning and corresponding changes in test performance when comparing HEBO with and without early stopping (Figure 14b). Each point in the scatter plot represents a single HPO run, with the x-axis denoting the change in overtuning and the y-axis the change in test performance – both computed such that positive values indicate worse outcomes for HEBO with early stopping. We observe a clear positive correlation between the two quantities, suggesting that reductions in overtuning achieved through early stopping tend to coincide with improved test performance. However, this relationship is not uniformly beneficial. While a substantial number of runs fall into the lower-left quadrant, indicating that early stopping reduces overtuning and improves test performance, there are also numerous instances in the upper-right quadrant where early stopping could not decrease overtuning yet harmed generalization (because we stopped too early). Moreover, the majority of points are concentrated near the origin, indicating that early stopping often has only a minor effect. These results confirm that early stopping can mitigate overtuning in some cases, leading to better generalization, but it does not consistently yield improvements and may even be detrimental.



(a) ECDF of the difference in test performance of the final incumbent for HEBO vs. HEBO with early stopping.



(b) Scatter plot of the difference in test performance of the final incumbent and the differences in final overtuning for HEBO vs. HEBO with early stopping.

Figure 14: Visualizations of the differences in test performance of the final incumbent and the differences in final overtuning for HEBO vs. HEBO with early stopping based on the *reshuffling* study (Nagler et al., 2024). 5-fold CV with ROC AUC as performance metric.

F Computational Details

As stated in Section 5 and Section 6 we rely on various published works that conducted HPO runs and published this data. With the exception of the HEBO runs with early stopping à la Makarova et al. (2022) as analyzed in Section 6 we did not run any new experiments. For these HEBO runs we used the code base of the *reshuffling* study (Nagler et al., 2024) released under MIT License. We estimate our total compute time for the HEBO with early stopping experiments to be roughly 0.63 CPU years. Benchmark experiments were run on an internal HPC cluster equipped with a mix of Intel Xeon E5-2670, Intel Xeon E5-2683 and Intel Xeon Gold 6330 instances. Jobs were scheduled to use a single CPU core and were allowed to use up to 16GB RAM. Total emissions are estimated to be an equivalent of roughly 345.96 kg CO2. The analyses reported in Section 5 and Section 6 require little computational power and were conducted on a personal computer.

We release all our code to perform the analyses reported in Section 5 and Section 6 via `https://anonymous.4open.science/r/paper_2025_overtuning-39CC/` under MIT License.