LMEMs for post-hoc analysis of HPO Benchmarking

Anton Geburek[1,*] **Neeratyoy Mallik**[1,*] **Danny Stoll**[1,*] **Xavier Bouthillier**[2] **Frank Hutter**[3,1]

[1]Machine Learning Lab, University of Freiburg
[2]Mila, Montreal
[3]ELLIS Institute Tübingen
[*]*Correspondence*: {gebureka,mallik,stolld}@cs.uni-freiburg.de

Abstract The importance of tuning hyperparameters in Machine Learning (ML) and Deep Learning (DL) is established through empirical research and applications, evident from the increase in new hyperparameter optimization (HPO) algorithms and benchmarks steadily added by the community. However, current benchmarking practices using averaged performance across many datasets may obscure key differences between HPO methods, especially for pairwise comparisons. In this work, we apply Linear Mixed-Effect Models-based (LMEMs) significance testing for post-hoc analysis of HPO benchmarking runs. LMEMs allow flexible and expressive modeling on the entire experiment data, including information such as benchmark meta-features, offering deeper insights than current analysis practices. We demonstrate this through a case study on the PriorBand paper's experiment data to find insights not reported in the original work.

1 Introduction

Hyperparameter Optimization (HPO) research has seen a surge in new benchmark contributions in recent times (Eggensperger et al., 2021; Pineda Arango et al., 2021; Wang et al., 2021; Pfisterer et al., 2022) that have led to improved HPO algorithm contributions too. This is a genuine attempt at making hyperparameter optimization (HPO) research an empirical and reproducible science, which is essential for the adoption of HPO in practice. The plethora of benchmarks can lead to large experimental data collected. The usual modus operandi is to use relative ranks per run instance to average the results across benchmarks for a seed, with the variance of this mean across seeds accounting for the uncertainty in relative ranks, thus compressing the experiment data into one easy-to-parse aggregated plot (Mallik et al., 2023). However, it is often observed that different problem instances can be solved best by different types of HPO algorithms (Eggensperger et al., 2021), akin to *No free lunch* (Wolpert and Macready, 1995).

In this work, we explore the application of model-based significance analysis to exploit the rich HPO experimental data from benchmarking runs. We believe that the relative performance of different HPO algorithms are strongly hierarchical in nature when considering different benchmarks or different HPO budget horizons. Moreover, we believe that finding sub-groups of benchmarks that capture different trends in relative performances can provide added insights.

The overall contribution of our work include: 1. Demonstrating application of LMEM-based significance analysis in HPO Benchmarking. 2. Preset LMEM-based model recipes for sanity checking experimental data and `Autorank`-like analysis. 3. Accounting for multiple benchmark metafeatures in pairwise comparison of HPO algorithms.
Our code: `https://github.com/automl/lmem-significance`.

2 Related Work and Background

LMEM-based significance testing has been brought to the attention of the Natural Language Processing community by (Riezler and Hagmann, 2022) to enable the joint analysis of both different

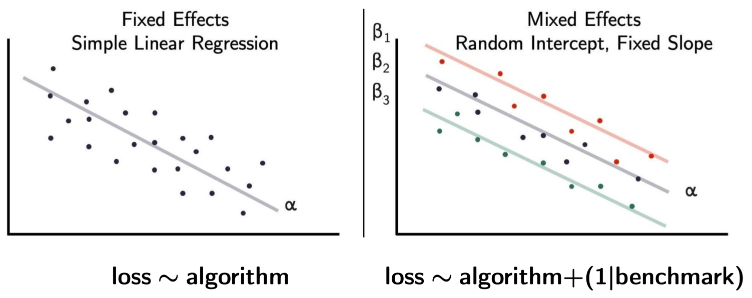


Figure 1: (*left*) Fixed effects are fully observed and typically noise-free, i.e., loss (y-axis) recorded against algorithms (x-axis); (*right*) Random effects assume samples to be from some random distribution within each specific group, as described by (1|benchmark) and the 3 lines representing 3 groups benchmarks. Image sourced under CC BY-SA 4.0.

data sets and meta-parameter settings. Such tests were recommended especially when there was a hierarchical structure within the data. LMEMs can account for both *fixed effects* and *random effects* (see, Figure 1). For HPO Benchmarking data, let M_0 : loss ∼ algorithm and M_1 : loss ∼ algorithm+(1|benchmark)[1], be two LMEM models fit on the same data. The Generalized Likelihood Ratio test (GLRT) compares the likelihood of the data given the model, to determine if algorithm can significantly effect the loss after accounting for random effects from different benchmark groups. This is in contrast to the commonly used Friedman test that doesn't assume normally distributed data groups or homogeneous variances (Demšar, 2006) but also cannot handle hierarchical relations in the data. The Wilcoxon test is also used to compare two algorithms over multiple benchmarks, collecting each significant win, tie, or loss as a ratio (Eggensperger et al., 2021). Dror et al. (2017) proposes to use a partial conjunction hypotheses to account for comparisons across multiple datasets, answering the question "does on algorithm A significantly outperform another algorithm B on at least u datasets out of N".

For methodological details, refer to Riezler and Hagmann (2022) and Appendix A.

3 Empirical setup

We directly demonstrate the application and utility of LMEM-based testing through a case study on real HPO Benchmarking data. We use experiment data from Mallik et al. (2023) containing runs of more than 5 different HPO algorithms, on more than 30 benchmark instances, repeated for 50 different seeds. For a focused study, we look at the main hypothesis and result in Mallik et al. (2023), that shows PriorBand to be better than HyperBand under different expert prior quality scenarios (see, Figure 6). Subsequently, our focus would be on the comparison of Random Search (RS), HyperBand (HB) and PriorBand (PB) over the benchmark instances comprising of *good* (at25) and *bad* expert prior input. To serve as a simple baseline simulating current standard practice, we use Autorank[2] as the baseline to perform Friedman test on the same experiment data. We use Critical-Difference (CD) (see, Fig. 2) plots to show pairwise significance difference in performance.

4 Application

In this section, we apply the method discussed to experiment data from Mallik et al. (2023).

4.1 Drop-in replacement for Autorank

Figure 2 highlights how a simple LMEM model can be used to model the entire experimental data for the 3 algorithms of concern , to yield the exact conclusion as current standard practice would

[1] here, loss is the target, algorithm is the fixed effect and benchmark is the random effect

[2] https://sherbold.github.io/autorank/

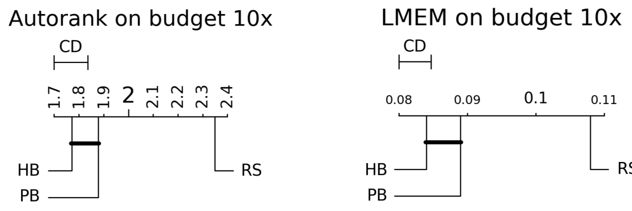
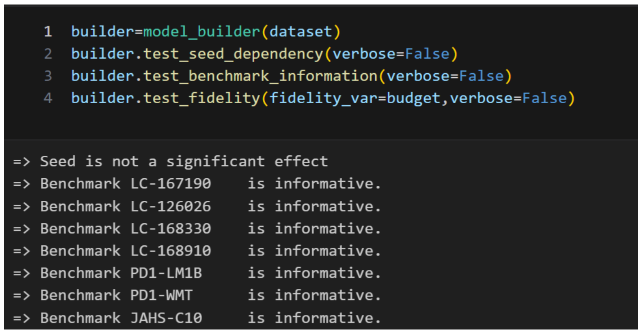


Figure 2: LMEMs can Autorank: (*left*) output of Autorank; (*right*) output of LMEMs on the same data with the simple model: loss ∼ algorithm.



```
builder=model_builder(dataset)
builder.test_seed_dependency(verbose=False)
builder.test_benchmark_information(verbose=False)
builder.test_fidelity(fidelity_var=budget,verbose=False)
```

```
=> Seed is not a significant effect
=> Benchmark LC-167190    is informative.
=> Benchmark LC-126026    is informative.
=> Benchmark LC-168330    is informative.
=> Benchmark LC-168910    is informative.
=> Benchmark PD1-LM1B     is informative.
=> Benchmark PD1-WMT      is informative.
=> Benchmark JAHS-C10     is informative.
```

Figure 3: Preset sanity check are run on the experiment data to conclude that there is no algorithm where the seed explains the performance variation. There is also no benchmark where there is no performance difference across algorithms. It was also found that the used budget should be used as an interaction effect for LMEM models on this data.

yield using Autorank. Here, the data seen by both setups is the same. Note, the difference in scale of the variance. This can be attributed to the difference in methodologies (refer to Appendix A). Figure 2 validates the use of model-based testing using LMEMs for the given data.

4.2 Sanity checks

Given that most HPO benchmarking runs will share a common set of metadata[3], LMEM models can be predefined given a data format to construct a sequence of LMEM models, that can be executed in a sequential decision-tree or workflow, checking for simple hypotheses. These are designed to catch potential erroneous algorithms, uninformative benchmarks, or buggy results from a vast collection of benchmarking runs. Some examples include: i) Is any algorithm performance explained by seeds ii) Is there any benchmark that shows no variation across algorithms iii) Find a complex model (hierarchically deep) that explains the data the best (see, Appendix C).

Figure 3 is an example run on the PriorBand experiment data. This forms an early check into the veracity of the experimentation setup and components. The Appendix C gives more details and verifies such preset recipes by means of synthetic datasets simulating such scenarios to catch.

4.3 Leveraging benchmark metafeatures with significance testing

The ability of LMEMs to capture hierarchical relations in the data given an expressive model opens up the potential to analyze HPO benchmarking runs with various metafeatures in consideration. In our case, for each benchmark, we find if PriorBand is significantly better than HyperBand, equivalent, or worse. To achieve this, the user can select an LMEM model of their choice. Given this information, and the metadata information per benchmark (in this case, we know if a benchmark instance is a *good* (at25) prior or a *bad* prior), we test if the relative performance of HyperBand

[3]typically: algorithm name, benchmark name, HPO budget spent, seed info, miscellaneous information

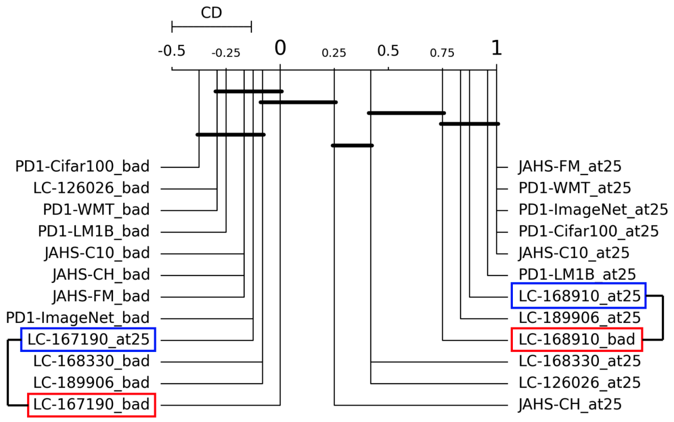


Figure 4: Clustering of benchmarks over prior qualities

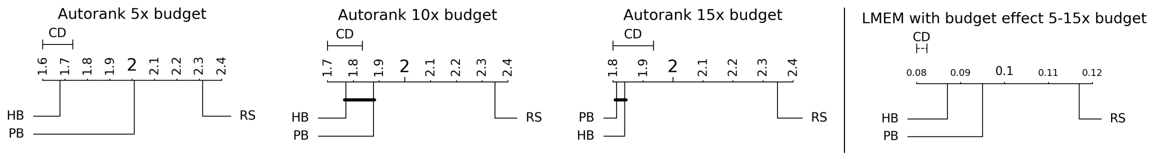


Figure 5: (*left*) `Autorank` at three different HPO budget horizons (5×, 10×, 15×); (*right*) LMEM trained on all available data from 5 − 15× budget, including the budget as a random effect: $\texttt{loss} \sim \texttt{algorithm} + (1|\texttt{budget}) + (1|\texttt{benchmark})$.

and `PriorBand` are significantly different or not for the two instances of the same benchmark task. As shown in Figure 4, there is significant performance difference between the two algorithms for the good-bad instances of each benchmark. This post-hoc analysis reveals two anamolous benchmarks that on further investigation appears to not have a *bad* enough expert prior designed (see, Figure 13). Mallik et al. (2023) did not consider this and such post-hoc analysis could have potentially prompted the authors to either omit these 2 benchmarks from result aggregation or redesign the priors. When benchmarking over large collection of benchmarks, we believe that such post-hoc analysis into which benchmarks contribute to the aggregated outcome is important for complete empirical understanding of results.

4.4 Further extensions and applications

The analysis from the previous section can be further expanded to a set of metafeatures. Forward-selection could determine the metafeature(s) that when included as a random effect(s) explains the experiment data the best. Looking into these subsets can provide useful information. Similarly, using the HPO budget spent as an effect can bring out nuances of how algorithms perform given an HPO budget window. We explore this briefly in Figure 5 but note that it still requires work to be the go-to analysis choice for *anytime* performance.

5 Conclusion

In this work, we propose and demonstrate the flexible application of LMEM-based significance testing in the context of HPO Benchmarking. More specifically, we show how LMEMs allow the modeling of potential hierarchical patterns in the benchmarking data, by accounting for random effects that can be attributed to different benchmark problems. We also show case how the HPO budget could be modeled as a fixed effect and allow for a novel compressed anytime performance analysis using a single CD plot. Moreoever, LMEMs offer the use of metafeatures for deeper analysis into relative performance of algorithms on subsets of benchmarks. LMEM-based methods such as these offer both the HPO researcher and practitioner to construct diverse hypotheses and obtain a different perspectives on their experiment data. Given standard data, our open-sourced Python package offers off-the-shelf recipes for the analysis presented in this work.

6 Limitations

In this paper, we aim to demonstrate the rich potential of applying flexible LMEM-based testing on standard HPO benchmarking data. We propose using these to supplement and aid existing analysis methods. To be the gold standard of significance testing for HPO, this warrants a longer and more scientific discussion.

While linear mixed-effects models (LMEMs) offer a powerful and flexible approach, their training and testing can be significantly slower than standard methods, especially for complex models or large datasets. This is partly due to our current implementation using the `pymer4` package, which relies on R in the background. We're actively working on a full Python implementation to address both speed and user-friendliness.

However, the flexibility of LMEMs also comes with a potential downside: the ease of modifying formulas can lead to misuse by inexperienced users. To mitigate this, we recommend using only basic formulas, employing the GLRT (Generalized Likelihood Ratio Test) to validate new effects, consulting with statisticians for complex models, and fully reporting all details when using LMEMs for significance testing in publications. It's important to acknowledge that the complexity of the model fit and potential limitations in the available data might lead to incorrectly rejecting the null hypothesis (essentially, finding a false difference). When in doubt, a simple ANOVA may be initially preferred.

Relatedly, we lack a statistical power analysis and did not extend the application of LMEM-based significance testing further than what is proposed by Riezler and Hagmann (2022) to include any control over Type II error. Further guidelines could be provided based on works such as the one of Matuschek et al. (2017).

7 Broader Impact

This paper applies an existing method in a post-hoc manner on existing benchmarking runs and, therefore, has no direct impact beyond the interested HPO researchers and practitioners. However, there is a possibility that the insights gained through the contributions of this paper can lead to more directed experiments, potentially saving computational resources and energy.

Acknowledgements. FH is a Hector Endowed Fellow at the ELLIS Institute Tübingen and acknowledges the financial support of the Hector Foundation. AG, NM, DS and FH acknowledge funding by the state of Baden-Württemberg through bwHPC, the German Research Foundation (DFG) through grant numbers INST 39/963-1 FUGG and 417962828, and the European Union (via ERC Consolidator Grant Deep Learning 2.0, grant no. 101045765), TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Funded by
the European Union

References

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. 7:1–30.

Dror, R., Baumer, G., Bogomolov, M., and Reichart, R. (2017). Replicability analysis for natural language processing: Testing significance with multiple datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486.

Eggensperger, K., Müller, P., Mallik, N., Feurer, M., Sass, R., Klein, A., Awad, N., Lindauer, M., and Hutter, F. (2021). HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In Vanschoren and Yeung (2021).

Mallik, N., Hvarfner, C., Bergman, E., Stoll, D., Janowski, M., Lindauer, M., Nardi, L., and Hutter, F. (2023). PriorBand: Practical hyperparameter optimization in the age of deep learning. In *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*.

Matuschek, H., Kliegl, R., Vasishth, S., Baayen, H., and Bates, D. (2017). Balancing type i error and power in linear mixed models. *Journal of memory and language*, 94:305–315.

Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., and Bischl, B. (2022). YAHPO Gym – an efficient multi-objective multi-fidelity benchmark 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.

Pineda Arango, S., Jomaa, H., Wistuba, M., and Grabocka, J. (2021). HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In Vanschoren and Yeung (2021).

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-Plus.* Springer. ISBN 0-387-98957-0.

Riezler, S. and Hagmann, M. (2022). *Validity, Reliability, and Significance: Empirical Methods for NLP and Data Science.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114.

Vanschoren, J. and Yeung, S., editors (2021). *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks.*

Wang, Z., Dahl, G., Swersky, K., Lee, C., Mariet, Z., Nado, Z., Gilmer, J., Snoek, J., and Ghahramani, Z. (2021). Pre-trained Gaussian processes for Bayesian optimization. *arXiv:2207.03084v4 [cs.LG].*

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

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]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

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

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)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [N/A]

   (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]

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [N/A]

   (e) Did you report the statistical significance of your results? [N/A]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A]

   (g) Did you compare performance over time and describe how you selected the maximum duration? [N/A]

   (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)? [N/A]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [N/A]

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)? [No] The URL will be released as part of the non-anonymized version of the paper.

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

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

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [N/A]

   (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]

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

   (a) Did you cite the creators of used assets? [Yes]

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes]

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

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]

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

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]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

A LMEMs and GLRTs overview

A.1 Short version

This likelihood-based testing approach trains an LMEM on the experimental data and estimates the mean of each algorithm based on the model's parameters. The results are normal distributions that can be compared via statistical tests, here we employ pairwise Tukey HSD tests. The model itself is used to capture additional information, disregarded in classical testing. We include the respective benchmark, seed, fidelity, and meta-features like the prior quality of each result. LMEMs are especially fit for this use case as they model inherent hierarchies in the data through the use of *random effects.* A variable like benchmark as *random effect* is assumed to be not fully observed but a sample from a zero-mean random distribution with an internally estimated variance-covariance matrix. As a result, they account for any variation caused by e.g. a different benchmark, effectively unifying the data. The approach therefore seeks to produce general results beyond what statistical tests normally could.
Additionally, two LMEMs M_0 and M_1 (with likelihoods l_0 and l_1 and numbers of parameters k_0 and k_1) can function as representations of hypotheses to be compared with the Generalized Likelihood Ratio Test (GLRT). As the models are fit by maximum likelihood, their likelihoods are normally distributed, producing a χ^2-testing distribution.

$$2 \log \frac{l_0}{l_1} \sim \chi^2_{k_0 - k_1}$$

A.2 Linear Mixed Effect Models

This work showcases the use of Linear Mixed Effect Models (LMEMs) for a model-based significance testing approach. These models have long been in use for their effectiveness at handling grouped data and within-group correlation, which is why they are now applied to HPO settings (Pinheiro and Bates, 2000). Briefly explained LMEMs extend the simple Linear Model ((1)) by additional components.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Lambda}_\theta) \tag{1}$$

In LMEMs, the vector β is called *fixed effects* to contrast the second, *random effects*-vector $\mathbf{b}$, which has its own design-matrix $\mathbf{Z}$.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \tag{2}$$

Contrary to $\boldsymbol{\beta}$, the random vector is assumed to consist only of samples from a normal distribution, similar to the error vector, instead of fully and reliably observed values, such that:

$$\mathbf{b} \sim \mathcal{N}(0, \boldsymbol{\psi}_\theta) \tag{3}$$

LMEMs now optimize the fixed effects matrix $\mathbf{X}$ directly, while the random effects matrix $\mathbf{Z}$ is determined from the distribution of $\mathbf{b}$, by estimating its variance-covariance matrix $\boldsymbol{\psi}_\theta$ through likelihood-maximization.
Now assume a model using the algorithm as fixed and the benchmark as a random effect ((4)). It has a grand intercept μ and for each algorithm a an effect v_a that gets activated by its indicator function $\mathbb{I}_a$. The same counts for the random effect benchmark, with its effects v_b and indicator function $\mathbb{I}_b$.

$$Y = \mu + \sum_{a \in algorithms} v_a * \mathbb{I}_a + \sum_{b \in benchmarks} v_b * \mathbb{I}_b + \epsilon_{residual} \tag{4}$$

This is logically extended to an ordinal variable x, where the sum over indicator functions is replaced by $v_x * x$ and interaction effects, with double sums and $v_{xy} * \mathbb{I}_{x \wedge y}$. This can introduce levels of hierarchy, which is an essential use case of LMEMs. As an example, for a categorical meta-parameter (or fidelity) p we can introduce a fixed effect v_p and an interaction effect v_{ap} to retrieve individual performances under each value of p:

$$Y = \mu + v_p \sum_{a \in algorithms} \left(v_a * \mathbb{I}_a + \sum_{p \in parameter} v_{ap} * \mathbb{I}_p \right) + \epsilon_{residual} \tag{5}$$

A.3 Estimated Marginal Means and Tukey-HSD

Estimated Marginal Means. Having built the model, we use *Estimated Marginal Means* (EMMs) to generate the testing data. In this process, an EMM grid is generated containing all unique values of all effects. Over this EMM grid, we can calculate the mean and standard error for each algorithm, again using the variance-covariance matrix $\boldsymbol{\psi}_\theta$. The resulting means and standard errors are then used as the basis for the actual significance test.
This is a unique property of model-based testing methods because we do not test on the data itself but rather use it as response variable for the model. This has considerable benefits, as it alleviates any restrictions on the distributional properties of the data like being normally distributed or having homogeneous variances. Any data can train the model while the model parameters themselves asymptotically follow a normal distribution, as they have been obtained by maximizing the likelihood (Demšar, 2006).

Pairwise comparisons with Tukey HSD. This understanding allows for a simple t-test for pairwise comparisons of each algorithm. As recommended by Riezler and Hagmann (2022) we use the *Tukey HSD test* to control the per-experiment Type-I error, as it is well suited for larger numbers of algorithms, contrary to the *Bonferroni correction.* The Tukey HSD test is a multiple testing corrected t-test introduced by Tukey (1949). We obtain the test statistic from the distance of two algorithms' estimated means and their common standard error:

$$q_s = \frac{|m_1 - m_2|}{SE_{m_1,m_2}} \tag{6}$$

The critical value q^* of this distribution is retrieved from the *studentized range distribution*, depending on the elected Type I error rate, the number of algorithms k and the number of observations per algorithm N_a, which give the degrees of freedom per algorithm as $df_a = N_a - k$. Using the cumulative studentized range distribution we can determine the p-value for this comparison of algorithms.

CD-Diagrams. From the Tukey HSD test we obtain an individual p-value for each comparison. This is ideal for constructing a CD-Diagram, a method introduced by Demšar (2006), using the Estimated Means as algorithm ranks. Importantly, we also obtain an individual *Critical Difference* for each comparison, which is why we extended the CD-Diagram to show the range from smallest to largest Critical Difference above.

A.4 Generalized Likelihood Ratio Testing

These LMEMs can in principle be extended by any possible factor but this comes at the cost of complexity and ability to generalize. To optimize this trade-off, we follow the approach of Riezler et al. to employ the *Generalized Likelihood Ratio Test* (GLRT) to compare models, specifically one with (M_1) against one without (M_0) the effect in question (Riezler and Hagmann, 2022). The GLRT compares the two models' likelihood l_1 and l_0 (Equation A.4) which in turn are both approximately normal, as they come from a maximum likelihood optimization. Therefore their ratio is χ^2 distributed with $k_0 - k_1$ degrees of freedom for k_i parameters of model M_i (Riezler and Hagmann, 2022).

$$2 \log \frac{l_0}{l_1} \sim \chi^2_{k_0 - k_1}$$

Using the critical value from this distribution, we can decide on which effects to add per the significance of the resulting p-value of the GLRT. A significant p-value lets us reject the null hypothesis that both models perform the same and, when additionally $l_0 > l_1$, conclude that the effect increases the model's power significantly.

B Data from `PriorBand`

In this section, we refer to all the relevant information from the original `PriorBand` work (Mallik et al., 2023) that supplements our case study of applying LMEM-based significant testing.

B.1 Subset considered

Figure 6 shows the primary result of concern in our case study. We want to understand the role of the different benchmarks, the quality of the default prior for that benchmark and how `HyperBand` and `PriorBand` differ in performance on them.

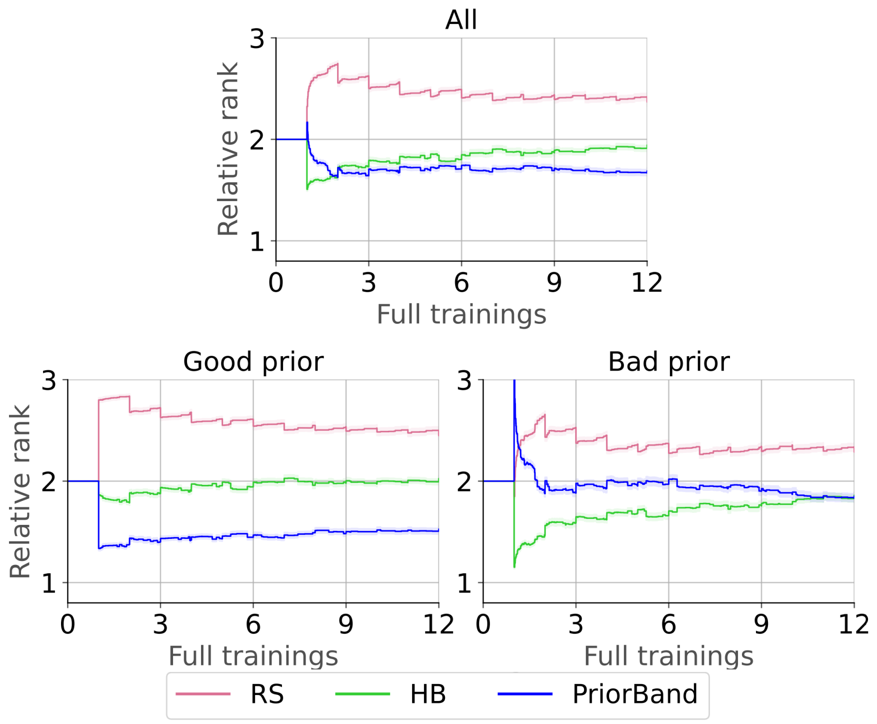
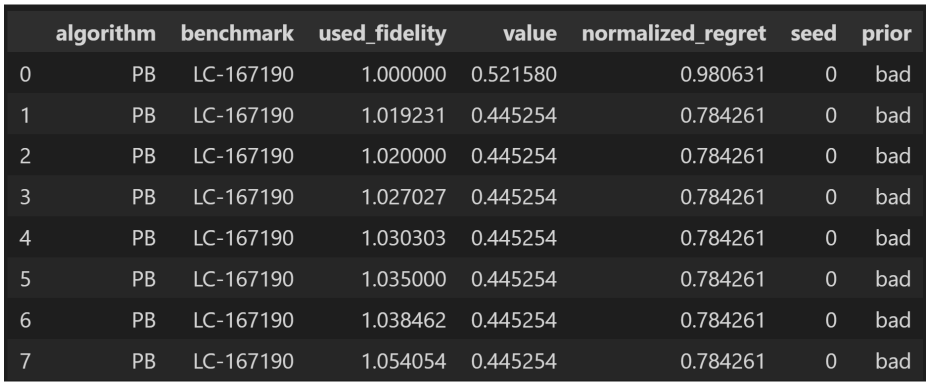


Figure 6: **Relative Ranks plot from the PriorBand Paper**: On the top: Relative ranks under both good (at25) and bad priors combined. On the bottom: Performance under individual priors. (Figure sourced from Mallik et al. (2023))

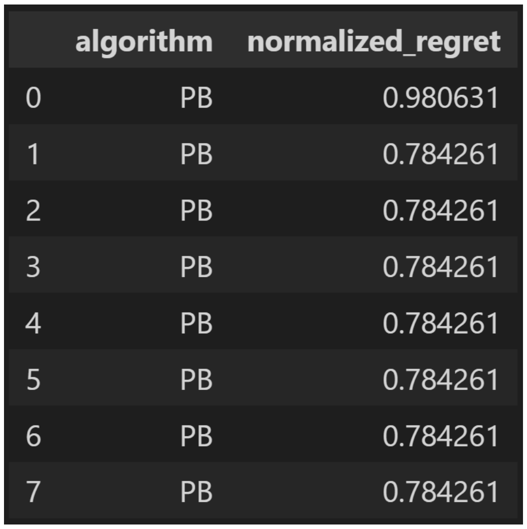
B.2 Metadata of experiment data

In this section we illustrate the structure of the HPO Benchmarking data we inherited. The original data from (Mallik et al., 2023) is shown in Figure 7. For Section 4.1 we use only the information about the loss and algorithm (Figure 8). For anytime performance analysis, we additionally include information about the training budget (Figure 9). In the sanity checks (Section 4.2), we also include the seed to analyze for seed dependencies (Figure 10). Finally in the benchmark clustering (Section 4.3), we include the loss, algorithm, benchmark, budget and prior quality (Figure 11).



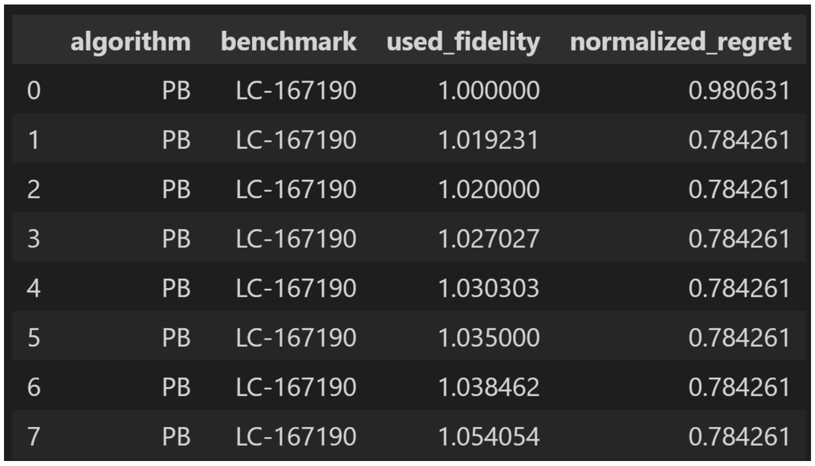
	algorithm	benchmark	used_fidelity	value	normalized_regret	seed	prior
0	PB	LC-167190	1.000000	0.521580	0.980631	0	bad
1	PB	LC-167190	1.019231	0.445254	0.784261	0	bad
2	PB	LC-167190	1.020000	0.445254	0.784261	0	bad
3	PB	LC-167190	1.027027	0.445254	0.784261	0	bad
4	PB	LC-167190	1.030303	0.445254	0.784261	0	bad
5	PB	LC-167190	1.035000	0.445254	0.784261	0	bad
6	PB	LC-167190	1.038462	0.445254	0.784261	0	bad
7	PB	LC-167190	1.054054	0.445254	0.784261	0	bad

Figure 7: **Excerpt from the PriorBand paper's data** (Mallik et al., 2023)
The data contains information on the loss and algorithm and on which benchmark, seed, fidelity, and prior it was acquired on.



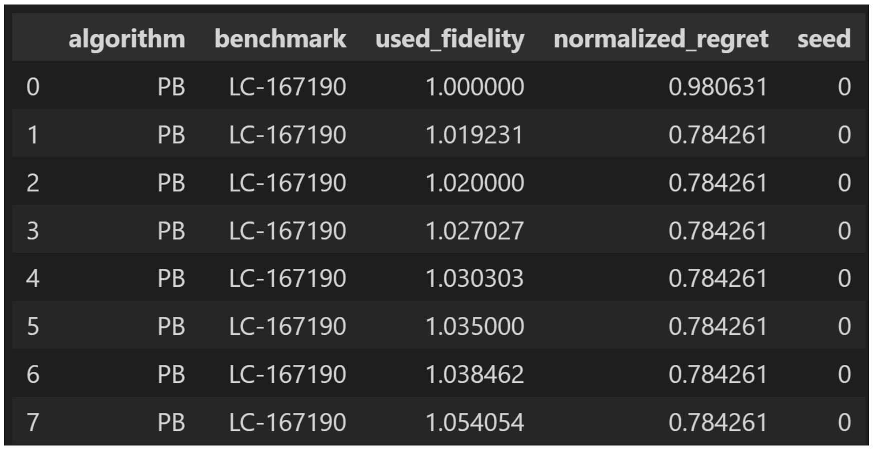
	algorithm	normalized_regret
0	PB	0.980631
1	PB	0.784261
2	PB	0.784261
3	PB	0.784261
4	PB	0.784261
5	PB	0.784261
6	PB	0.784261
7	PB	0.784261

Figure 8: **Data used for comparing `LMEMs` to `Autorank`**
The data contains the loss and the algorithm.



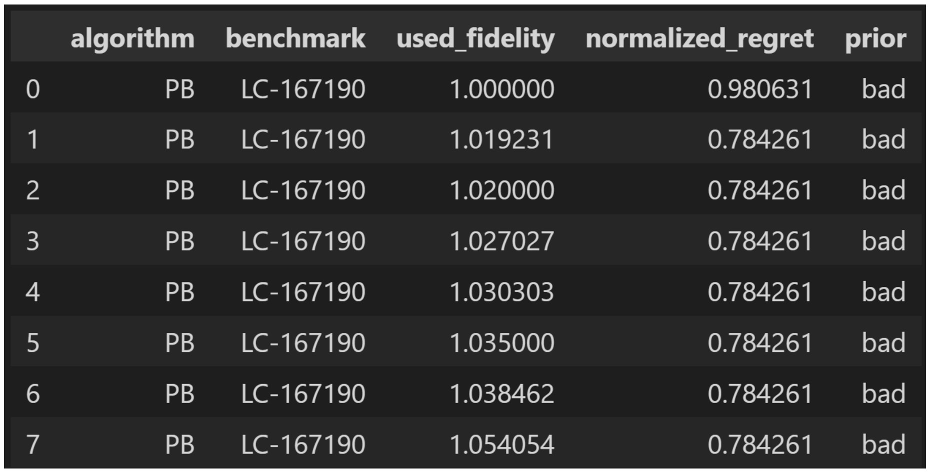
	algorithm	benchmark	used_fidelity	normalized_regret
0	PB	LC-167190	1.000000	0.980631
1	PB	LC-167190	1.019231	0.784261
2	PB	LC-167190	1.020000	0.784261
3	PB	LC-167190	1.027027	0.784261
4	PB	LC-167190	1.030303	0.784261
5	PB	LC-167190	1.035000	0.784261
6	PB	LC-167190	1.038462	0.784261
7	PB	LC-167190	1.054054	0.784261

Figure 9: **Data used for anytime analysis**
The data contains the loss, algorithm and training budget.



	algorithm	benchmark	used_fidelity	normalized_regret	seed
0	PB	LC-167190	1.000000	0.980631	0
1	PB	LC-167190	1.019231	0.784261	0
2	PB	LC-167190	1.020000	0.784261	0
3	PB	LC-167190	1.027027	0.784261	0
4	PB	LC-167190	1.030303	0.784261	0
5	PB	LC-167190	1.035000	0.784261	0
6	PB	LC-167190	1.038462	0.784261	0
7	PB	LC-167190	1.054054	0.784261	0

Figure 10: **Data used in the sanity checks**
The data contains information on the loss and algorithm and on which benchmark, seed, and fidelity it was acquired on.

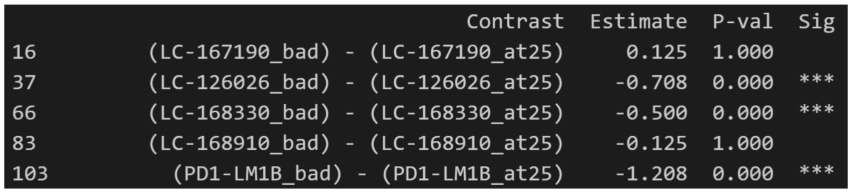


	algorithm	benchmark	used_fidelity	normalized_regret	prior
0	PB	LC-167190	1.000000	0.980631	bad
1	PB	LC-167190	1.019231	0.784261	bad
2	PB	LC-167190	1.020000	0.784261	bad
3	PB	LC-167190	1.027027	0.784261	bad
4	PB	LC-167190	1.030303	0.784261	bad
5	PB	LC-167190	1.035000	0.784261	bad
6	PB	LC-167190	1.038462	0.784261	bad
7	PB	LC-167190	1.054054	0.784261	bad

Figure 11: **Data used for benchmark clustering**
The data contains information on the loss and on which benchmark, fidelity, and prior it was acquired on.

B.3 Supporting plots borrowed

Figure 13 shows the distribution of error given a prior configuration to sample around. The different violins indicate different prior inputs. These particular 2 benchmarks from Mallik et al. (2023) were marked in our analysis to be the odd ones under expected behaviour and trends with other benchmarks. On looking deeper, it turned out these 2 benchmarks have bimodal *bad* prior distributions. With one of the modes at a similar performance level as the good prior mode. Thus, suggesting that these benchmarks could have been dropped from the aggregation results or the prior qualities should have been reassessed.



```
                                  Contrast  Estimate  P-val  Sig
16         (LC-167190_bad) - (LC-167190_at25)     0.125  1.000
37         (LC-126026_bad) - (LC-126026_at25)    -0.708  0.000  ***
66         (LC-168330_bad) - (LC-168330_at25)    -0.500  0.000  ***
83         (LC-168910_bad) - (LC-168910_at25)    -0.125  1.000
103          (PD1-LM1B_bad) - (PD1-LM1B_at25)    -1.208  0.000  ***
```

Figure 12: **Benchmark compared under good and bad priors via clustering**: Two benchmarks, *LC-Bench 167190* and *LC-Bench 168910* show no significant difference between their prior variants.

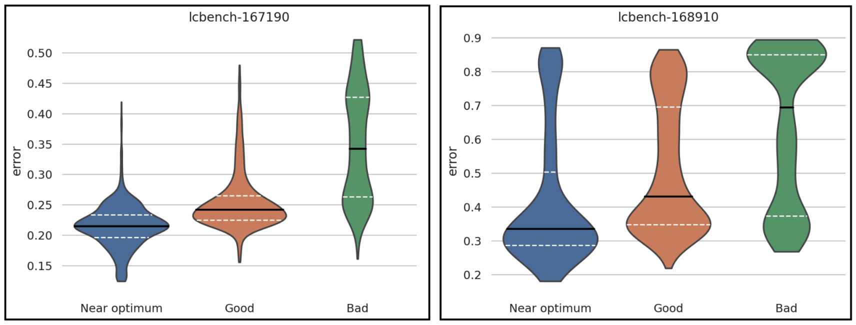


Figure 13: **Violin plots from the PriorBand Paper**: Both benchmarks that showed insignificant differences in prior qualities turn out to have strongly bimodal *bad* prior distributions with good quality secondary modes.

C Sanity checks on synthetic data

In Section 4.2 we present several sanity checks that the PriorBand data passed. We now create synthetic datasets, to show other results of these checks.

Seed-Independency. We have generated a small synthetic dataset of three algorithms at 50 seeds that share the same mean (2.5) and variance (0.55), but where algorithm A-1 is influenced by the seed (Figure 14). A GLRT compares two models, with (Equation 7) and without seed effect (Equation 8), to see whether an algorithm performance might depend on the seed chosen.

$$\textbf{loss} \sim \textbf{algorithm} \tag{7}$$

$$\textbf{loss} \sim \textbf{algorithm} + (\textbf{0} + \textbf{algorithm}|\textbf{seed}) \tag{8}$$

The second model fits a random effect for each seed and algorithm. If this does not increase the significance of the model, the GLRT will reject it meaning we cannot make a significant connection between seed and algorithm performance. In this case, it accepts the complex model, so we look for large variances in this random effect. These variances show exactly which algorithm is affected.

Benchmark relevance. For each benchmark, we train a model with (Equation 9) and without (Equation 10) algorithm effect, so assuming either significant or insignificant differences between the algorithms.

$$\textbf{loss} \sim \textbf{1} \tag{9}$$

$$\textbf{loss} \sim \textbf{algorithm} \tag{10}$$

Again we show this using a test set where the algorithm's mean performance depends on the Benchmark:

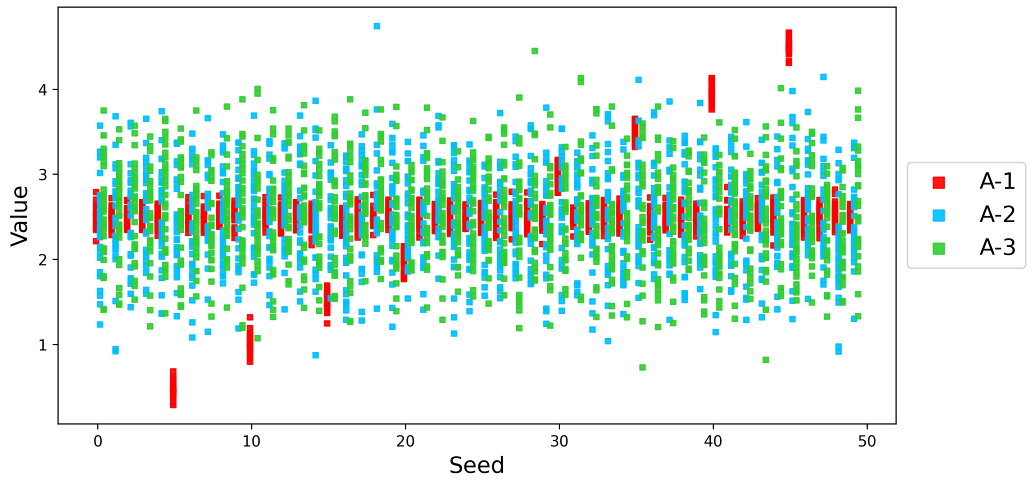
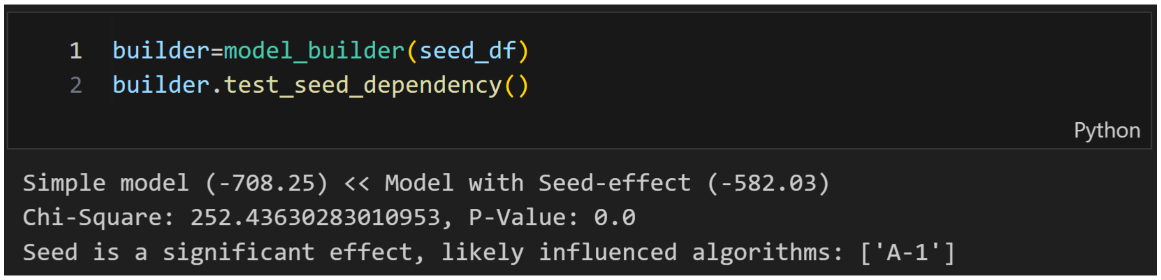


Figure 14: **Seed-dependent synthetic dataset** We create a synthetic dataset with three algorithms where for some seeds algorithm A-1 uses $0.1 * seed$ as mean.



```
builder=model_builder(seed_df)
builder.test_seed_dependency()
```

```
Simple model (-708.25) << Model with Seed-effect (-582.03)
Chi-Square: 252.43630283010953, P-Value: 0.0
Seed is a significant effect, likely influenced algorithms: ['A-1']
```

Figure 15: **Test on Seed dependency** We use a GLRT to compare a model that includes the seed as a factor against one that does not (Simple model). The p-value is significant and the seed model is more likely than the simple one, so the seed is a significant effect. In the second step, we analyze the effect's variances and correctly determine algorithm A-1 to be the only affected algorithm.

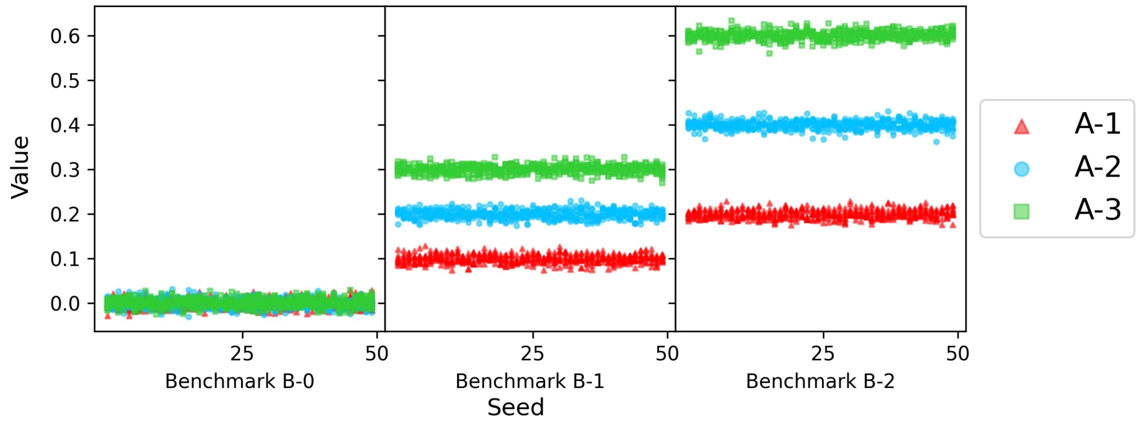
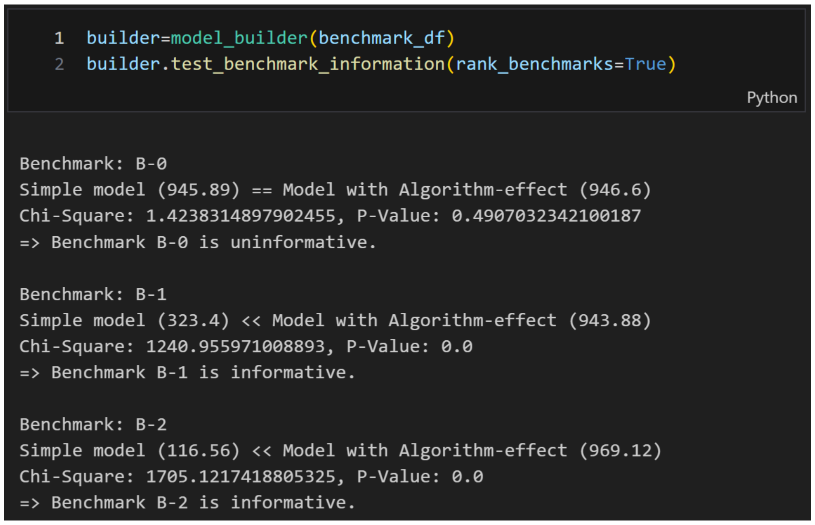


Figure 16: **Multi-Benchmark dataset**: For Benchmark B-0, all algorithms perform similar, for B-1 and B-2, performances vary.

Additionally (but more compute-intensively) we can rank the benchmarks according to their relevance, by using a model with individual algorithm-benchmark random effects (Figure 11) and again looking at the variance of these effects, where higher variance corresponds to larger performance differences.

$$\textbf{loss} \sim \textbf{(0 + benchmark|algorithm)} \tag{11}$$



```python
builder=model_builder(benchmark_df)
builder.test_benchmark_information(rank_benchmarks=True)
```

```
Benchmark: B-0
Simple model (945.89) == Model with Algorithm-effect (946.6)
Chi-Square: 1.4238314897902455, P-Value: 0.4907032342100187
=> Benchmark B-0 is uninformative.

Benchmark: B-1
Simple model (323.4) << Model with Algorithm-effect (943.88)
Chi-Square: 1240.955971008893, P-Value: 0.0
=> Benchmark B-1 is informative.

Benchmark: B-2
Simple model (116.56) << Model with Algorithm-effect (969.12)
Chi-Square: 1705.1217418805325, P-Value: 0.0
=> Benchmark B-2 is informative.
```

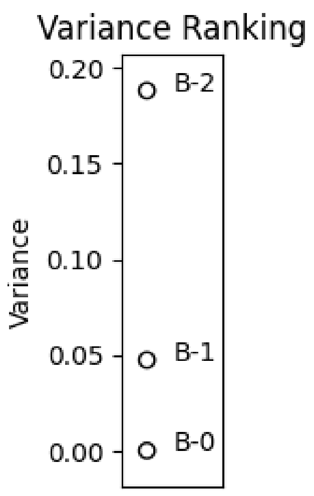


Figure 17: **Benchmark relevance test** Left: Individual tests, Right: Variance ranking

As Figure 17 shows, benchmark B-0 is uninformative as in the algorithms do not vary significantly within the benchmark and the ranking suggests the largest performance gap for B-2, which corresponds to our data.

Budget relevance. LMEMs can natively integrate the HPO budget into the model training process. To test whether a fidelity or other Meta-Parameter variation helps understand the data, we propose

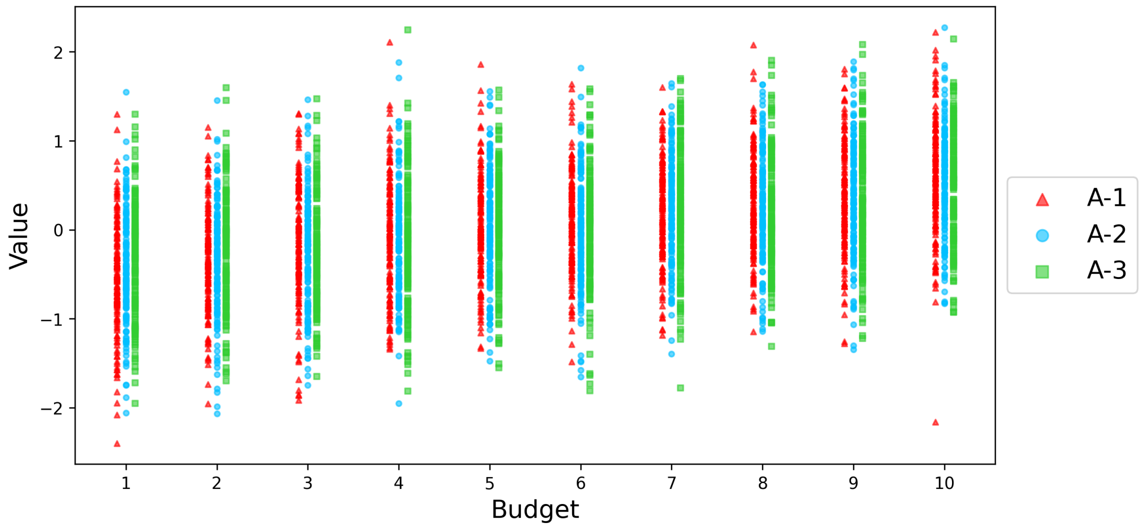


Figure 18: **Budget dependent dataset**: The training budget loosely affects the mean of each algorithm.

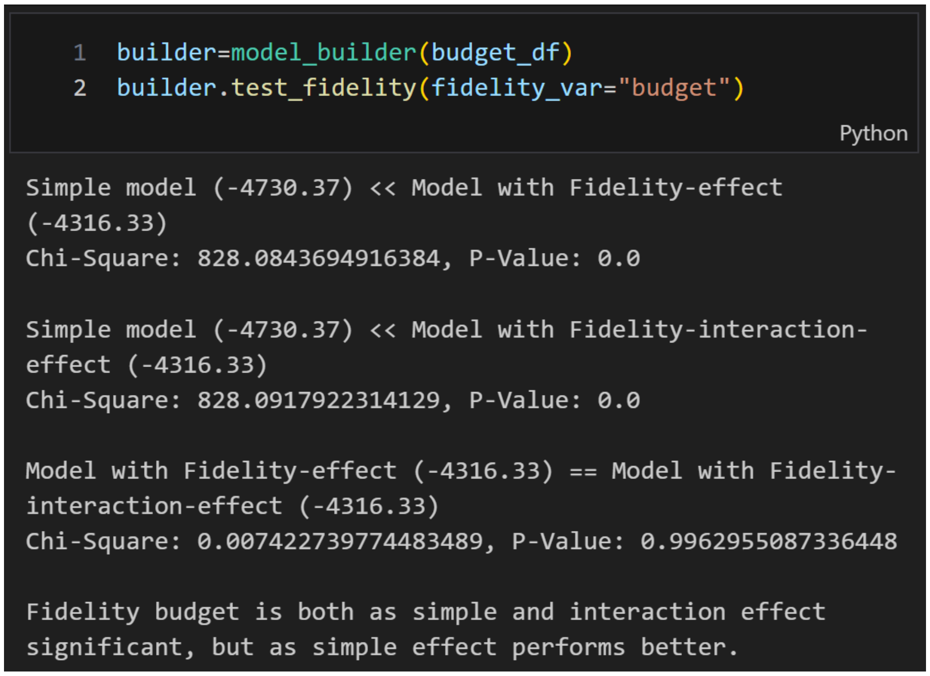
a set of GLRTs that compares a simple model (Equation 12), a model with fixed fidelity effect

(Equation 13) and a model with interaction effect (Equation 14) to each other.

$$\textbf{loss} \sim \textbf{algorithm} + (\textbf{1}|\textbf{benchmark}) \tag{12}$$

$$\textbf{loss} \sim \textbf{algorithm} + \textbf{budget} + (\textbf{1}|\textbf{benchmark}) \tag{13}$$

$$\textbf{loss} \sim \textbf{algorithm} + \textbf{algorithm:budget} + (\textbf{1}|\textbf{benchmark}) \tag{14}$$



```python
builder=model_builder(budget_df)
builder.test_fidelity(fidelity_var="budget")
```

```
Simple model (-4730.37) << Model with Fidelity-effect
(-4316.33)
Chi-Square: 828.0843694916384, P-Value: 0.0

Simple model (-4730.37) << Model with Fidelity-interaction-
effect (-4316.33)
Chi-Square: 828.0917922314129, P-Value: 0.0

Model with Fidelity-effect (-4316.33) == Model with Fidelity-
interaction-effect (-4316.33)
Chi-Square: 0.007422739774483489, P-Value: 0.9962955087336448

Fidelity budget is both as simple and interaction effect
significant, but as simple effect performs better.
```

Figure 19: **Testing budget effect** After comparing models with and without the effects, we conclude that budget is significantly improving the model both as a simple and as an interaction effect. However, as an interaction effect introduces higher complexity, the simple effect performed better of the two.